INSTITUTO TECNOLÓGICO DE SONORA

MANUAL DE PRÁCTICAS PARA LA PROGRAMACIÓN DE MICROCONTROLADORES PIC's DE LA FAMILIA 16FXXX

TESIS
QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN ELECTRÓNICA

PRESENTA

CARLOS ARMANDO ARMENTA BUITIMEA

CD. OBREGÓN, SONORA **JULIO DE 2007**

CONTENIDO

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

RESUMEN

El propósito de este trabajo es auxiliar al estudiante en el proceso de aprendizaje de programación de los microcontroladores PIC's, de la familia 16FXXX, en particular de los modelos PIC16F628 y PIC16F87X.

Los microcontroladores PIC's, son herramientas muy poderosas para el diseño de proyectos, los cuales están desplazando a los microcontroladores de propósito general, debido a que existe un microcontrolador PIC que cuenta con los recursos necesarios para cada aplicación.

A continuación se presenta este documento, el cual consta de cuatro capítulos, un apéndice y tres anexos, donde se describen paso a paso las herramientas para la realización de programas en el lenguaje ensamblador, así como la simulación y finalmente la depuración utilizando el software ***PROTEUS***. Se da una introducción del lenguaje de programación en ensamblador para PIC's y su ventaja en el desarrollo de trabajos de diseño digital con esta herramientas tan utilizadas por muchas empresas.

En el capítulo I, se realiza una introducción donde se plantean los antecedentes, planteamiento del problema, objetivo general, objetivo particular, justificación y las delimitaciones de este trabajo.

Dentro del capítulo II, se expone la metodología para llevar a cabo este manual; así como su estructura, el entorno de desarrollo donde se realizaron cada una de las prácticas, los objetivos específicos de cada práctica, como también el software que se utilizó.

Dentro del capítulo III, se muestran los resultados que se obtuvieron durante el desarrollo de este trabajo.

En el capítulo IV se exponen las conclusiones a las que se llegó al término de este trabajo, también se realiza un listado de la bibliografía consultada como apoyo para este trabajo.

Dentro del Apéndice A, se presenta el manual completo propuesto, el cual contiene diez prácticas. Con las que se espera que el estudiante sea capaz de realizar aplicaciones con estos microcontroladores.

El Anexo A, se exponen las características generales de los microcontroladores PIC16F62X y PIC16F87X, así como también la descripción detallada del repertorio de instrucciones de los microcontroladores en cuestión.

El Anexo B, presenta el programa utilizado para enviar el resultado obtenido en la práctica 7 y 8 de éste manual.

Por último en el Anexo C, presenta la conexión del circuito integrado MAX232, utilizada para realizar la comunicación serial con la PC.

DEDICATORIAS

A mis padres

Ramón Armenta y Armida Buitimea, por guiarme por el camino correcto para alcanzar mis objetivos personales y darme el apoyo necesario para realizarme como profesionista, gracias por darme todo su amor y comprensión, los quiero mucho.

A mis hermanos

Juan Ramón, Sergio Rubén, Leonardo y José Gustodio, gracias hermano por todo lo que me han ofrecido y darme la oportunidad de realizar mis estudio, gracias por todos estos momentos que hemos vivido y por el gran cariño que nos tenemos.

A mi hermana

Dora Martina, gracias por todo su cariño y ayuda para lograr mis objetivos, por estar siempre a mi lado y por darme tu comprensión y apoyo. Te quiero mucho hermana.

A mis amigos

Juan Carlos (Compare), Benjamín, Roberto (Rata), Carlos (Charly), Alonso (Tweny), Julián (Secón), Alonso (Loncho), Agustín Jaime (Tilin), Smith que han estado conmigo desde hace mucho tiempo ofreciéndome su amistad que es tan valiosa para mi, como también por sus comentarios y críticas, tremendamente honestos.

AGRADECIMIENTOS

A Dios

En primer lugar, por prestarme esta vida, por darme salud, perseverancia y los grandes deseos de lucha para seguir adelante. Gracias por enseñarme que sin tropiezos no hay experiencias.

A mis padres

Deseo dar las gracias de forma muy especial a Ramón Armenta y Armida Buitimea, por su gran esfuerzo que han hecho durante toda su vida para darme lo mejor. Sin su cariñosa presencia y ánimos, este trabajo no hubiese merecido la pena.

A mis hermanos

Dora Martina, Juan Ramón, Sergio Rubén, Leonardo y José Gustodio, por su gran apoyo durante toda esta lucha inagotable y ser parte importante durante mi formación profesional.

Al instituto Tecnológico de Sonora

Agradezco, por todo lo que obtuve en ella, conocimientos, experiencias, amigos, compañeros, permitir formarme como profesionista y el orgullo se pertenecer a esta institución. Gracias.

A mi asesor

Agradezco, al M.C. Eduardo Romero Aguirre, por el tiempo y la paciencia que dedico para ayudarme y guiarme en la revisión de este trabajo, también por la enseñanza que tuve la oportunidad de recibir en las materias que me impartió.

I INTRODUCCIÓN

1.1 Antecedentes

Fue a principios de los años 70 cuando apareció en el mercado electrónico, el circuito integrado denominado microprocesador, el cual revolucionó el campo de la electrónica digital y analógica de una manera rapidísima y eficaz. Se implementaron numerosos sistemas de control e instrumentación industrial en torno a los microprocesadores, que sin duda alguna se imponían, no solamente en precio sino además en rendimiento y nuevas posibilidades a los sistemas hasta entonces existentes.

Los microprocesadores funcionan básicamente como una unidad de procesamiento y control de datos. Para llevar a cabo todas las operaciones que son capaces de realizar, necesitan disponer en su entorno de una serie de elementos. Estos componentes auxiliares del microprocesador son entre otros, las memorias RAM, las memorias PROM, las memorias EPROM, los periféricos de entradas/salidas, etc.

Los fabricantes de este tipo de microcircuitos, dándose cuenta de todo esto, desarrollaron componentes que engloban en un sólo chip gran parte de estos elementos, es decir, resuelven en un sólo componente las funciones propias del microprocesador, además de las necesidades de memorias de programa, memoria de datos, elementos de entrada/salida para comunicarse con el exterior, elementos temporizadores, etc. Estos nuevos microcircuitos especializados generalmente en aplicaciones industriales, constituyen lo que llamamos microcontroladores.

Lógicamente, a medida que elevamos el nivel de exigencia o demanda de nuestro nuevo microcircuito, se eleva su complejidad.

Actualmente, los fabricantes de microcontroladores tienden a incorporar en sus diseños, cada vez mayor potencialidad y más controladores externos, cuyo uso es exclusivo para tareas determinadas y concretas.

Los microcontroladores PIC de Microchip, abarcan una gran variedad de modelos, que sin duda, permiten a los usuarios elegir el más interesante o apropiado para los intereses de cada proyecto.

1.2 Planteamiento del problema

Debido a la reestructuración del plan 1995, los programas analíticos fueron modificados con el fin de actualizar y mejorar los cursos impartidos, a nivel licenciatura en el “Instituto Tecnológico de Sonora”.

El enfoque del plan 1995 es el método tradicional de enseñanza, el cual cambia por un método mucho más ambicioso como lo es el enfoque de competencias.
Una vez implementado este enfoque, se analizó la manera de actualizar el microcontrolador objeto de estudio.

La materia de “Sistemas Digitales III”, dentro de su programa analítico incorporaba el estudio de programación del microcontrolador de ***MOTOROLA*** de la familia MC68HC11, el cual en la actualidad se encuentra descontinuado, por lo que es difícil encontrarlo comercialmente.

En la actualidad, en el mercado electrónico existen dispositivos con la misma fiabilidad y a un menor costo; estos microcontroladores son de la empresa ***MICROCHIP*** denominados PIC’s. De los cuales existe un dispositivo para cada aplicación específica. Debido a que la tendencia de la electrónica que es aprovechar al máximo los recursos tanto en hardware y software.

1.3 Objetivo general

Diseñar un manual de prácticas didácticas relativas a la programación de los periféricos de los microcontroladores PIC’s de la familia 16F628 y 16F87X, que pueda ser empleado como manual del laboratorio para la materia de Sistemas Digitales III.

1.4 Objetivo particular

- Describir los pasos necesarios para elaborar un programa y aplicaciones con los MCU’s PIC’s de Microchip.
- Explicar el uso de cada una de las herramientas de hardware y software necesarias para programar los microcontroladores PIC’s.
- Exponer en forma clara y sencilla la forma de configurar adecuadamente cada uno de los periféricos internos de los microcontroladores PIC16F628 y PIC16F87X.
- Mostrar el procedimiento para configurar las diversas interrupciones con las que cuenta los microcontroladores PIC16F62X y PIC16F87X.

1.5 Justificación

Los microcontroladores PIC's, hoy en día se encuentran en todos los sitios, desde el ratón de la computadora hasta el control de los frenos ABS del automóvil, pasando por el televisor, el ascensor, la lavadora, los juguetes de los niños, el teléfono móvil, etc. Por lo que son un buen prospecto para su estudio.
Es muy importante que los ingenieros en electrónica dominen el diseño e implementación de circuitos digitales, bajo este enfoque de control con los microcontroladores PIC's. Por su gran peculiaridad, como lo es su flexibilidad para su adaptación a las diferentes tareas que pueden llevar a cabo, estos microcontroladores.

Se prestó especial atención en la materia de "Sistemas Digitales III", ya que esta materia ofrece una gran formación para el mundo laboral, por este motivo es muy importante para cualquier futuro ingeniero en electrónica.

1.6 Delimitaciones

- Debido a los tiempos asignados a las sesiones de laboratorio, el manual contará de sólo diez prácticas.
- Cada una de las prácticas se dimensionará para una duración de 2 horas aproximadamente ya que es el tiempo de duración de una sesión de laboratorio.
- Las prácticas están centradas en los microcontroladores PIC16F62X y PIC16F87X únicamente.
- El dispositivo programador que se contempla en las prácticas es un "clon" del JDM programmer.

II METODOLOGÍA

Tomando como referencia que el semestre consta de 15 sesiones, en este trabajo se enfocará en distribuir las 10 prácticas durante ese lapso. La filosofía fundamental detrás de cada una de las prácticas es guiar al alumno a realizar programas en lenguaje ensamblador para verificar el correcto funcionamiento de los diferentes recursos y/o periféricos internos, con que cuentan estos microcontroladores para el desarrollo de aplicaciones.

Los programas que se realizaron son los más usuales dentro del funcionamiento de los microcontroladores. Para ello se toma en cuenta que una sesión del laboratorio tiene una duración de 2 horas, este es el punto de partida para sintetizar al máximo los programas propuestos para cada práctica, así se tiene el tiempo suficiente para la realización y comprensión, de cada una de los temas.

El contenido de cada una de las prácticas es de acuerdo al orden gradual en el aprendizaje de cada una de las distintas operaciones de los componentes internos de los microcontroladores. Así, se comienza con el manejo de entorno de

programación MPLAB y se termina con la programación de PIC's en el lenguaje de alto nivel.

2.1 Estructura del manual

El orden y secuencia de cada una de las prácticas se hizo de tal forma que armonizara con el programa del curso de la materia de *Sistemas Digitales III*. De tal forma, que cualquier concepto teórico visto en el aula de clases quedaría comprendido y reforzado con su correspondiente sesión de laboratorio.

A continuación se presenta la calendarización sugerida para poder usar este manual (ver tabla 2.1).

Tabla 2.1 Estructura del manual.

SESIÓN	PRÁCTICA	DURACION	NOMBRE
1	------	2 hrs	Presentación de las políticas del curso y criterios de evaluación
2	1	2 hrs	Introducción al software MPLAB IDE de MICROCHIP
3	2	2 hrs	Introducción al software de simulación PROTEUS "ISIS"
4	3	2 hrs	Bits de configuración y puertos paralelos del PIC16F628
5	4	2 hrs	Direccionamiento indirecto y teclado matricial
6	5	2 hrs	Interrupciones externas del PIC16F628
7	6	2 hrs	Convertidor A/D del PIC16F87X
8 9	7	2 hrs 2 hrs	Temporizadores programables del PIC16F628
10 11	8	2 hrs 2 hrs	Módulos CCP del PIC16F628

12 13	9	2 hrs 2 hrs	Modulo USART del PIC16F628 en modo asíncrono
14	10	2 hrs	Programación para PIC's en lenguaje C
15	Entrega de calificación		

2.2 Elección de las herramientas de software

La filosofía que se ha venido siguiendo es elegir herramientas de uso popular. Así se seleccionó el programa ***MPLAB*** de ***MICROCHIP***, que es la herramienta más difundida a nivel mundial, este software cuenta con todas herramientas necesarias para la creación de proyectos, y puede ser descargado e instalado sin costo alguno de la pagina de ***MICROCHIP***.

En este software fueron editados, compilados y depurados todos los programas, Así también se comprobó el correcto funcionamiento a nivel circuital por medio del software de simulación ***PROTEUS***. Una vez concluido el proceso de diseño se procedió a grabar el código máquina en la memoria interna del PIC, utilizando el software ***IC-PROG***.

A continuación se hace un listado del software utilizado:

- ✓ Software computacional Mplab IDE v7.40
- ✓ Proteus 6 profesional
- ✓ Pic C compiler 3.163.
- ✓ Ic-Prog version 1.05A

2.3 Estructura general de las prácticas

Durante el desarrollo de este trabajo se identificó que el formato a seguir en cada una de las prácticas era fundamental para cumplir los objetivos planteados en ellos.

Así pues, en cada práctica se comienza con una breve introducción teórica para luego plantear los objetivos que se persiguen; a continuación, se enlistan los materiales y equipo requerido para cada práctica y las actividades previas que el estudiante debe cumplir.

Posteriormente viene la parte de desarrollo, donde el alumno experimentalmente validará y reforzará sus conocimientos teóricos. Finalmente cada práctica plantea actividades extraclase a manera de preguntas que deberá investigar, para complementar aspectos teóricos más específicos.

Todo lo anterior queda plasmado en los siguientes puntos que todas las prácticas contienen:

- Introducción al tema de estudio
- Objetivo
- Material empleado
- Pre-reporte
- Desarrollo
- Actividades complementarias

III RESULTADOS

Para la validación de las prácticas, se realizó el siguiente proceso (ver figura 3.1). El primer paso fue hacer una lluvia de ideas entre el asesor y el tesista, para proponer el primer diseño en borrador, de cada uno de los experimentos, los cuales son editados por el tesista, para posteriormente ser minuciosamente revisadas por el asesor, quien es el encargado de proponer nuevas mejoras, así como nuevos diseños didácticos. Esta persona es quien toma la decisión de rechazar y/o aceptar el diseño propuesto. En caso de ser rechazada se procede al rediseño del borrador con las correcciones propuestas para la práctica, este proceso no tiene un límite, sólo es liberado hasta cumplir con las espectativas de tema en cuestión.

Una vez cumplido el proceso de revisión, y haber obtenido el visto bueno del asesor. El siguiente paso es someter las prácticas a un proceso un poco más delicado ya que son aplicadas a una tercera persona, la cual no cuenta con los conocimientos acerca del tema.

Esta tercera persona es la que brinda la retroalimentación de cada una de las prácticas, para efectuar las correcciones pertinentes, tanto en el aspecto didáctico como en los tiempos que se manejarían para el desarrollo de cada una de ellas. Si la práctica se rechaza, esta se regresa para ser modificada y volver al punto anterior donde se le aplica una rigurosa revisión a las nuevas correcciones las cuales son supervisadas por el asesor, para regresar de nuevo a la revisión de la tercera persona. En caso de ser aprobada con éxito, la práctica se da por concluida.

Por último, se añaden actividades complementarias a cada una de las prácticas ya validadas, las cuales sirven para reforzar detalles específicos acerca de lo aprendido en la práctica.

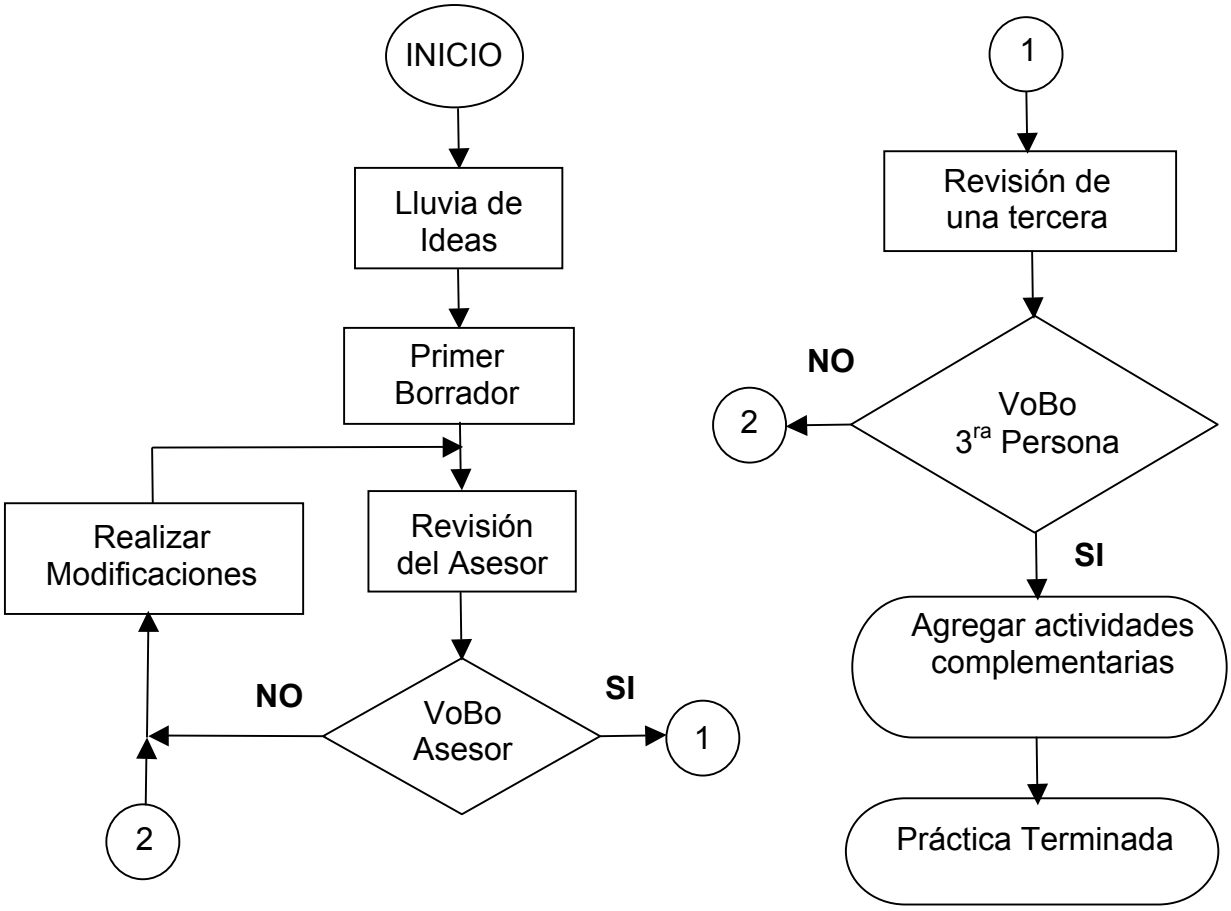


Figura 3.1 Proceso de validación de prácticas

IV CONCLUSIONES

Este manual presenta una guía para desarrollar distintas aplicaciones de las diferentes herramientas y/o recursos internos con los que cuentan los microcontroladores PIC's, que pueden ser accedidas y programadas también en el lenguaje C. En la actualidad muchas empresas realizan su diseño e implementación de proyectos, en los cuales incluyen estos microcontroladores.

Una de las ventajas es la facilidad con la que se pueden modificar los programas empleados en cada práctica y adaptarlos a necesidades particulares del usuario.

En este trabajo se muestra una forma de programar los recursos y/o herramientas, de una forma básica y algunos pasos para realizar diseños de control un poco más completos y funcionales. Se deberá tener en cuenta que no es la única forma de trabajar con los recursos internos de los microcontroladores y que otras personas podrán resolver el mismo problema del microcontrolador mediante otra secuencia de programación o simplemente con el diseño más simplificado.

Durante y después de la realización de este tipo de trabajo se adquiere una gran experiencia en lo que respecta al sentido didáctico que debe tener cada una de las sesiones aquí presentadas y se espera obtener una respuesta satisfactoria por parte de las personas que lo utilicen como guía para el diseño de proyectos utilizando la familia de microcontroladores PIC16FXXX.

Al final de este trabajo se cumplieron los objetivos de este manual para cada una de las prácticas aquí propuestas, desde el diseño de ellas, los problemas que se propusieron y los objetivos individuales de cada una para la correcta comprensión.

REFERENCIAS

[1] www.microchip.com

[2] www.labcenter.co.uk/index_uk.htm

[3] www.IC-Prog.com

[4] MICROCHIP, PIC16f62X Data sheet, USA, 2003

[5] MICROCHIP, PIC16f87X Data sheet, USA, 2003

[6] Cuenca, E. Martín. eta al, Microcontroladores PIC, La solución en un solo CHIP, Paraninfo, España 2001.

[7] ANGULO, J. M. eta al, Microcontroladores PIC, Diseño Práctica de Aplicaciones, Primera Parte: PIC16F84, McCGraw-Hill, España 1999.

[8] ANGULO, J. M. eta al, Microcontroladores PIC, Diseño Práctica de Aplicaciones, Segunda Parte: PIC16F87X, McCGraw-Hill, España 1999.

[9] SMITH, D. W., PIC in Practice, Newnes, USA, 2002

[10] Barnett, Cox & O'Cull. Embedded C Programming and the Microchip PIC Thomson-Learning, 1er Edition, 2004.

[11] M. en C. Liñan G. José L. & González G. Alain C. Estructura de datos y Lenguaje C, Arbol de la vida, Edo. De México 1995.

ANEXO A

CARACTERÍSTICAS GENERALES DEL MICROCONTROLADOR PIC16F62X Y PIC16F87X

MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

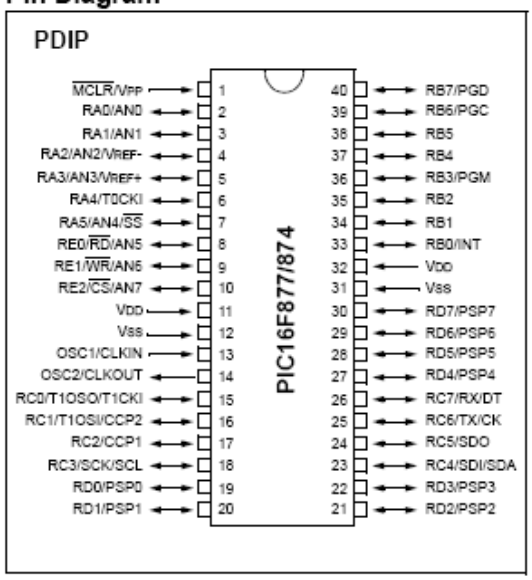
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 μA typical @ 3V, 32 kHz
  - < 1 μA typical standby current

Pin Diagram

PDIP



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

MICROCHIP

PIC16F62X

FLASH-Based 8-Bit CMOS Microcontrollers

Devices Included in this Data Sheet:

- PIC16F627
- PIC16F628

Referred to collectively as PIC16F62X

High Performance RISC CPU:

- Only 35 instructions to learn
- All single cycle instructions (200 ns), except for program branches which are two-cycle
- Operating speed:
  - DC - 20 MHz clock input
  - DC - 200 ns instruction cycle

Device	Memory		
	FLASH Program	RAM Data	EEPROM Data
PIC16F627	1024 x 14	224 x 8	128 x 8
PIC16F628	2048 x 14	224 x 8	128 x 8

- Interrupt capability
- 16 special function hardware registers
- 8-level deep hardware stack
- Direct, Indirect and Relative addressing modes

Peripheral Features:

- 16 I/O pins with individual direction control
- High current sink/source for direct LED drive
- Analog comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (V_{REF}) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Timer1: 16-bit timer/counter with external crystal/clock capability
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Capture, Compare, PWM (CCP) module
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Universal Synchronous/Asynchronous Receiver/Transmitter USART/SCI
- 16 Bytes of common RAM

Special Microcontroller Features:

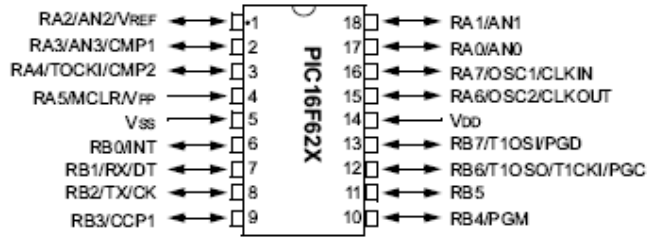
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Detect (BOD)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Multiplexed $\overline{MCLR}$-pin
- Programmable weak pull-ups on PORTB
- Programmable code protection
- Low voltage programming
- Power saving SLEEP mode
- Selectable oscillator options
  - FLASH configuration bits for oscillator options
  - ER (External Resistor) oscillator
    - Reduced part count
  - Dual speed INTRC
    - Lower current consumption
  - EC External Clock input
  - XT Oscillator mode
  - HS Oscillator mode
  - LP Oscillator mode
- In-circuit Serial Programming™ (via two pins)
- Four user programmable ID locations

CMOS Technology:

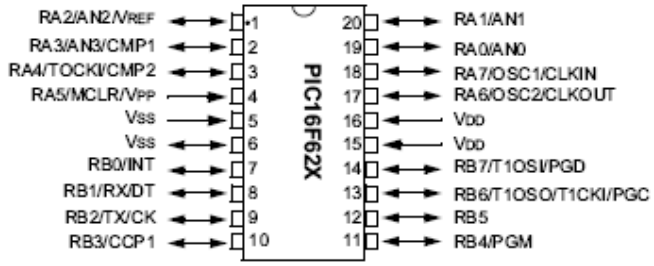
- Low power, high speed CMOS FLASH technology
- Fully static design
- Wide operating voltage range
  - PIC16F627 - 3.0V to 5.5V
  - PIC16F628 - 3.0V to 5.5V
  - PIC16LF627 - 2.0V to 5.5V
  - PIC16LF628 - 2.0V to 5.5V
- Commercial, industrial and extended temperature range
- Low power consumption
  - < 2.0 mA @ 5.0V, 4.0 MHz
  - 15 μA typical @ 3.0V, 32 kHz
  - < 1.0 μA typical standby current @ 3.0V

Pin Diagrams

PDIP, SOIC



SSOP



Device Differences

Device	Voltage Range	Oscillator	Process Technology (Microns)
PIC16F627	3.0 - 5.5	(Note 1)	0.7
PIC16F628	3.0 - 5.5	(Note 1)	0.7
PIC16LF627	2.0 - 5.5	(Note 1)	0.7
PIC16LF628	2.0 - 5.5	(Note 1)	0.7

Note 1: If you change from this device to another device, please verify oscillator characteristics in your application.

13.1 Instruction Descriptions

ADDLW	Add Literal and W
Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.

ADDWF	Add W and f
Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

ANDLW	AND Literal with W
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.

ANDWF	AND W with f
Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

BCF	Bit Clear f
Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

BSF	Bit Set f
Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b < 7
Operation:	skip if (f<b>) = 1
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', the next instruction is executed. If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2TCY instruction.

BTFSC	Bit Test, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

CALL	Call Subroutine
Syntax:	[*label*] CALL k
Operands:	0 ≤ k ≤ 2047
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
Description:	Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction.

CLRWDT	Clear Watchdog Timer
Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h → WDT 0 → WDT prescaler, 1 → $\overline{TO}$ 1 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

CLRF	Clear f
Syntax:	[*label*] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Description:	The contents of register 'f' are cleared and the Z bit is set.

COMF	Complement f
Syntax:	[*label*] COMF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	($\bar{f}$) → (destination)
Status Affected:	Z
Description:	The contents of register 'f' are complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f'.

CLRW	Clear W
Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.

DECF	Decrement f
Syntax:	[*label*] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination)
Status Affected:	Z
Description:	Decrement register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

DECFSZ	Decrement f, Skip if 0
Syntax:	[*label*] DECFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination); skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are decremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, then a NOP is executed instead making it a 2TCY instruction.

INCFSZ	Increment f, Skip if 0
Syntax:	[*label*] INCFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (destination), skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, a NOP is executed instead, making it a 2TCY instruction.

GOTO	Unconditional Branch
Syntax:	[*label*] GOTO k
Operands:	0 ≤ k ≤ 2047
Operation:	k → PC<10:0> PCLATH<4:3> → PC<12:11>
Status Affected:	None
Description:	GOTO is an unconditional branch. The eleven-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two-cycle instruction.

IORLW	Inclusive OR Literal with W
Syntax:	[*label*] IORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .OR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are OR'ed with the eight bit literal 'k'. The result is placed in the W register.

INCF	Increment f
Syntax:	[*label*] INCF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (destination)
Status Affected:	Z
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.

IORWF	Inclusive OR W with f
Syntax:	[*label*] IORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .OR. (f) → (destination)
Status Affected:	Z
Description:	Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

MOVF	Move f
Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (destination)
Status Affected:	Z
Description:	The contents of register f are moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register, since status flag Z is affected.

MOVLW	Move Literal to W
Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

MOVWF	Move W to f
Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Description:	Move data from W register to register 'f'.

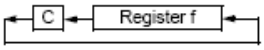
NOP	No Operation
Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Description:	No operation.

RETFIE	Return from Interrupt
Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None

RETLW	Return with Literal in W
Syntax:	[*label*] RETLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W); TOS → PC
Status Affected:	None
Description:	The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.

RLF — **Rotate Left f through Carry**

Syntax:	[*label*] RLF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is stored back in register 'f'.



SLEEP

Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT, 0 → WDT prescaler, 1 → $\overline{TO}$, 0 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	The power-down status bit, $\overline{PD}$ is cleared. Time-out status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.

RETURN — **Return from Subroutine**

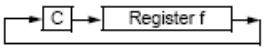
Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → PC
Status Affected:	None
Description:	Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.

SUBLW — **Subtract W from Literal**

Syntax:	[*label*] SUBLW k
Operands:	$0 \le k \le 255$
Operation:	k - (W) → (W)
Status Affected:	C, DC, Z
Description:	The W register is subtracted (2's complement method) from the eight-bit literal 'k'. The result is placed in the W register.

RRF — **Rotate Right f through Carry**

Syntax:	[*label*] RRF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.



SUBWF — **Subtract W from f**

Syntax:	[*label*] SUBWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - (W) → (destination)
Status Affected:	C, DC, Z
Description:	Subtract (2's complement method) W register from register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

ANEXO B

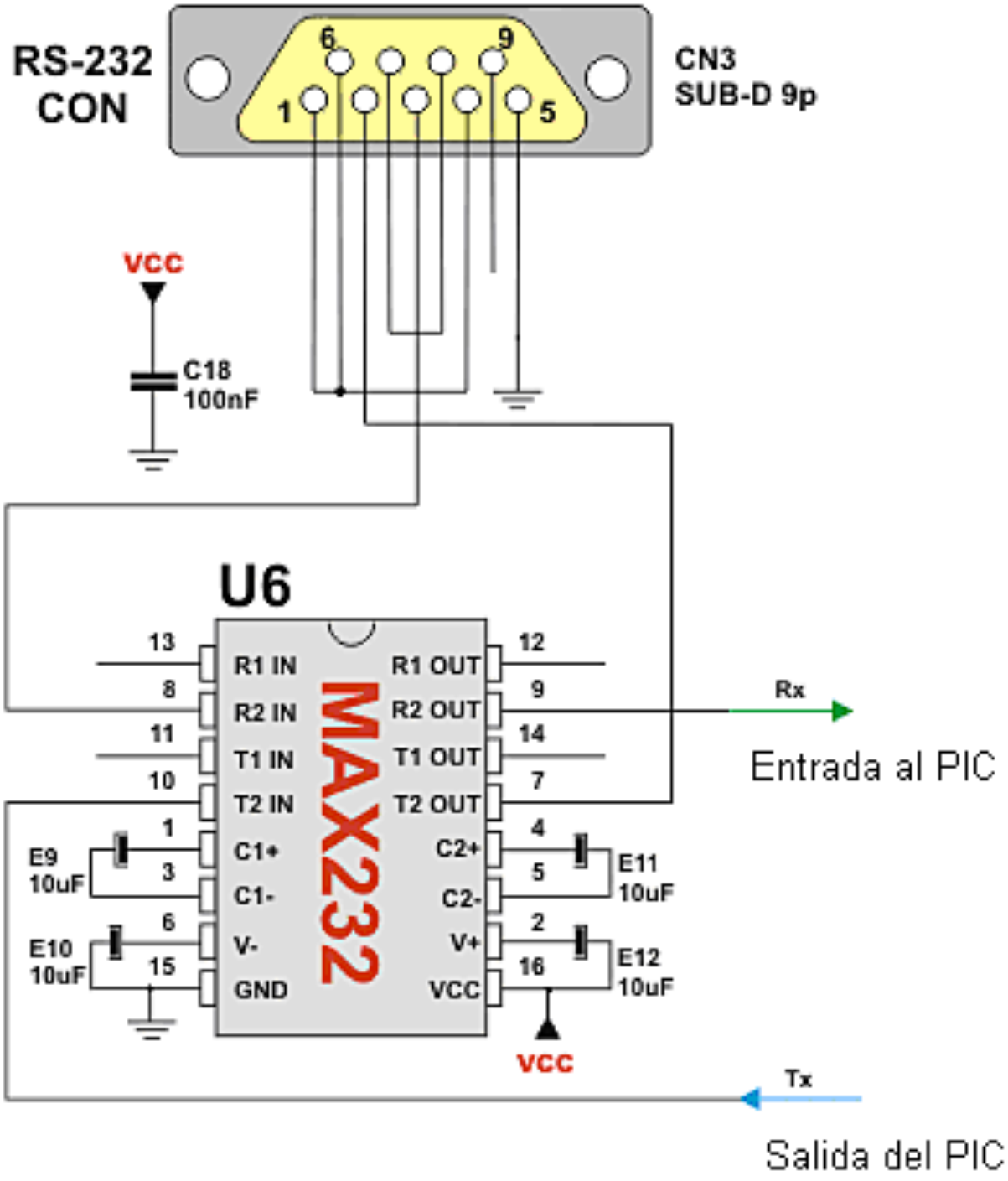
LIBRERÍA UTILIZADA PARA ENVIAR LOS DATOS CAPTURADOS POR COMUNICACIÓN SERIAL

```
;*******************************************************************************
; Programa que transmite por HYPERTERMINAL una cadena de datos
;*******************************************************************************
        CBLOCK                  ; Variables utilizadas
        msnib
        lsnib
        ENDC

;*******************************************************************************
REGR_CAR
        call        BANDRA
        movlw       .13         ; Transmite un LF (alimentador a la izquierda)
        call        ENVIA
        call        BANDRA

        call        BANDRA
        movlw       .10         ; Transmite un CR (retorno de carro)
        call        ENVIA
        call        BANDRA

        call        BANDRA
        movlw       .0          ; Transmite un 0 (alimentador al inicio)
        call        ENVIA
        call        BANDRA
        return

;*******************************************************************************
BANDRA
        bsf         STATUS,RP0      ; Conmuta a banco 1
        btfss       TXSTA,TRMT      ; Checas si el buffer de transmisión
        goto        $-1             ; esta habilitado.
        bcf         STATUS,RP0      ; Conmuta a banco 0
        return

;*******************************************************************************
;Subrutina que envía el byte en W por el puerto serie separándolos en sus
;respectivos códigos ASCII de sus dos nibbles Hexadecimales.
;*******************************************************************************
DATO
        movwf       msnib           ; Lees el byte msnib
        movwf       lsnib           ; Copia al lsnib
        swapf       msnib,f         ; Interrcambia nibbles en lsnib
        movlw       0x0F            ; Mascara para limpiar nibble alto
        andwf       msnib,f         ; Limpia nibble alto del msnib
        andwf       lsnib,f         ; Limpia parte baja del lsnib
        movf        msnib,w         ; Carga msnib en w
        call        ASC             ; Rutina que convierte en ASCII
```

```
        call    ENVIA           ; Para enviarlo por el puerto
        call    BANDRA          ; serie
        call    BANDRA
        movf    lsnib,w         ; Carga lsnib en w
        call    ASC             ; Manada llamar el ASCII
        call    ENVIA           ; Del valor del Registro
        call    BANDRA          ; y envia el dato
        call    BANDRA          ; Revisa la bandera
        clrf    msnib           ; Que este libre
        clrf    lsnib           ; Borra las variables
        clrf    TXREG           ; y el registro de transmisión
        return

;*******************************************************************************************
;
ASC
        Addwf   PCL,f           ; Calcular el código a retornar
        DT "0123456789ABCDF"    ; Saltando w instrucciones adelante

;*******************************************************************************************
;
; Subrutina que inicializa el puerto serie USART como transmisor a 9600 bauds,
;considerando un cristal de reloj de 4MHZ
;*******************************************************************************************
;
INIC_TX
        bcf     STATUS,RP1
        bsf     STATUS,RP0      ; Conmuta a banco 1
        bsf     TXSTA,BRGH      ; Habilita el bit BRGH=0
        movlw   D'25'           ; Valor para generar 9600 Bauds
        movwf   SPBRG           ; Limpia bit SYNC
        bcf     TXSTA,SYNC      ; para el Modo asíncrono
        bsf     TXSTA,TXEN      ; Habilitas transmisor
        bcf     STATUS,RP0      ; Conmuta a banco 0
        bsf     RCSTA,SPEN      ; Habilita Puerto Serie
        clrf    TXREG           ; Limpiar registro transmisor
        return

;*******************************************************************************************
;
;Subrutina que envia el byte guardando en W por el puerto serie
;*******************************************************************************************
;
ENVIA
        bsf     STATUS,RP0      ; Conmuta a banco 1
        btfss   TXSTA,TRMT      ; Checas si el buffer de
        goto    $-1             ; transmisión esta habilitado
        bcf     STATUS,RP0      ; Conmuta a banco 0
        movwf   TXREG           ; Guarda el dato en w
        return
```

ANEXO C

CONEXIÓN SERIAL RS-232 CONECTOR DB9


RS-232
CON
6
9
1
5
CN3
SUB-D 9p
VCC
C18
100nF
U6
MAX232
R1 IN
R2 IN
T1 IN
T2 IN
C1+
C1-
V-
GND
R1 OUT
R2 OUT
T1 OUT
T2 OUT
C2+
C2-
V+
VCC
13
8
11
10
1
3
6
15
12
9
14
7
4
5
2
16
E9
10uF
E10
10uF
E11
10uF
E12
10uF
VCC
Rx
Entrada al PIC
Tx
Salida del PIC