Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

MANUAL DE LABORATORIO DE MICROPROCESADORES PIC USO DE LENGUAJES DE PROGRAMACIÓN BASIC Y ASSEMBLER

Carlos Manuel Boche Leonardo
Asesorado por el Ing. Armando Alonso Rivera Carrillo

Guatemala, septiembre 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

MANUAL DE LABORATORIO DE MICROPROCESADORES PIC USO DE LENGUAJES DE PROGRAMACIÓN BASIC Y ASSEMBLER

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

CARLOS MANUEL BOCHE LEONARDO
ASESORADO POR EL ING. ARMANDO ALONSO RIVERA CARRILLO

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN ELECTRÓNICA

GUATEMALA, SEPTIEMBRE 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Oscar Humberto Galicia Núñez
VOCAL V	Br. Carlos Enrique Gómez Donis
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Byron Odilio Arrivillaga Méndez
EXAMINADOR	Ing. José Aníbal Silva de los Ángeles
EXAMINADOR	Ing. Romeo Neftalí López Orozco
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

MANUAL DE LABORATORIO DE MICROPROCESADORES PIC USO DE LENGUAJES DE PROGRAMACIÓN BASIC Y ASSEMBLER

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería de Mecánica Eléctrica, con fecha 14 de Julio 2010.

Carlos Manuel Boche Leonardo

Guatemala 5 de Febrero de 2018

Señor
Coordinador de Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Estimado Coordinador

Por este medio hago de su conocimiento que he finalizado la revisión del trabajo de graduación del estudiante Carlos Manuel Boche Leonardo, titulo "MANUAL DE LABORATORIO DE MICROPROCESADORES PIC USO DE LENGUAJES DE PROGRAMACION BASIC Y ASSEMBLER "Considerando que el mismo cumple a cabalidad de los objetos propuestos al momento de su aprobación.

Por lo que, lo remito a su persona para que sirva continuar con el tramite respectivo además que, tanto el autor como el suscrito en calidad de ASESOR NOMBRADO, somos responsables del contenido del trabajo de graduación.

Atentamente,

Ing. Armando Alonso Rivera Carrillo
ASESOR

Armando Alonso Rivera Carrillo
Ingeniero Electrónico
Colegiado No. 4265

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Guatemala, 19 de febrero de 2018

Señor Director
Ing. Otto Fernando Andrino Gonzalez
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC

Señor Director:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado: **MANUAL DE LABORATORIO DE MICROPROCESADORES PIC USO DE LENGUAJES DE PROGRAMACION BASIC Y ASSEMBLER**, desarrollado por el estudiante **Carlos Manuel Boche Leonardo,** ya que considero que cumple con los requisitos establecidos.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

ID Y ENSEÑAD A TODOS

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

Ing. Julio Cesar Solares Peñate
Coordinador de Electrónica

NIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

REF. EIME 06. 2018.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante; CARLOS MANUEL BOCHE LEONARDO titulado: MANUAL DE LABORATORIO DE MICROPROCESADORES PIC USO DE LENGUAJES DE PROGRAMACIÓN BASIC Y ASSEMBLER procede a la autorización del mismo.

Ing. Otto Fernando Andrino González

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 1 DE MARZO 2,018.

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

DTG. 344.2018

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **MANUAL DE LABORATORIO DE MICROPROCESADORES PIC USO DE LENGUAJES DE PROGRAMACIÓN BASIC Y ASSEMBLER,** presentado por el estudiante universitario: **Carlos Manuel Boche Leonardo,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Pedro Antonio Aguilar Polanco
Decano

Guatemala, septiembre de 2018

/gdech

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

ACTO QUE DEDICO A:

Dios	Por darme la oportunidad de ser mejor persona cada día.
Mi padre	Víctor Boche.
Mi Madre	Por apoyarme incondicionalmente.
Mi esposa	Matilde Lucrecia Martínez Aguirre mi amiga y compañera de estudio, gracias por el apoyo en los momentos difíciles.
Mi hija	Bianca Ximena, por ser una nueva inspiración a mi vida.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Mi casa de estudio, estás en mi corazón.
Facultad de Ingeniería	Por darme las herramientas para realizar mis sueños y ambiciones.
Mis amigos de la Facultad	Julio Morales, Cristian Gómez, Carlos Arreola, Hugo Dardón. Gracias por la amistad y el apoyo en cada curso o proyecto.
Mis amigos Ruta	Gracias Jose Villeda, Carlos Santos y Manuel Elias.
Mis compañeros de oficina	Carlos Franco, Marvin Perez, Samuel Bonilla, Selvin Brocks, Dieter Soto, Mario Velasquez, Enrique Chete y muchos otros que me cubrían las espaldas mientras terminaba de escribir mi trabajo de graduación.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

INDICE DE TABLAS

GLOSARIO

ADC Es la conversión de una señal análoga a valores digitales.

ASM Abreviatura de *Assembly Language*, lenguaje de bajo nivel que tiene palabras claves (nemotécnicos) para ordenar a un procesador realizar operaciones aritméticas entre un conjunto de datos o el traslado de información entre bancos de información.

Big Data Es el término que describe una gran cantidad de información almacenada, en la actual industria tecnológica se utiliza esta información para optimizar servicios. Cuanta más información se tenga, mejores cálculos predictivos de cualquier servicio se pueden realizar.

Bit Es el acrónimo de Valor Binario (*Binary Digit*). Es la medida de información digital más pequeña es un sistema de numeración basado únicamente en dos signos 1 o 0.

Bus I2C Es el bus de comunicación que se utiliza para la conexión entre periféricos externos y un microprocesador I2C es el acrónimo de circuito integrado (*Inter-Integrated Circuit).*

CAN	Protocolo de comunicación basado en una topología bus para la transmisión de mensajes en entornos distribuidos. Es el acrónimo de *Controller Area Network*. Una de sus ventajas es la inmunidad a las interferencias, reduce las conexiones por medio de cables ya que elimina las conexiones punto a punto.
Display	Es un conjunto de led´s colocados en una manera estratégica que ayuda a visualizar un valor digital, se considera un periférico de salida ya que permite mostrar un valor numérico al usuario.
LCD	Es una pantalla electrónica donde se muestra información de manera óptima ya que se pueden mostrar caracteres muy particulares con los kanjis (letras japonesas) es el acrónimo de *Liquid Cristal Display*,
Periféricos de E/S	Son los periféricos que se utilizan como entrada o salida según sea la necesidad del caso.
PIC	Es un integrado programable que internamente tiene distintos componentes como una unidad de almacenaje interno, unidad de procesamiento y puertos de entrada de información como también de salida. Es el acrónimo de *Program Integrated Circuit*

PWM	Tipo de modulación donde la información del mensaje es enviada en variaciones en el ancho de pulso de una señal ya predefinida. Es el acrónimo de *Pulse Width Modulation*.
USART	Protocolo de transmisión asíncrono. Es una comunicación serial punto a punto, utiliza un tren de 8 pulsos de datos, un pulso de inicio y otro de paro de información, es el acrónimo universal *asynchronous receiver-transmitter*.
Zigbee	Es el nombre de un conjunto de protocolos de comunicación inalámbrica su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías

RESUMEN

La finalidad de este trabajo de graduación es proporcionar al estudiante una herramienta para el aprendizaje de los microprocesadores *PIC´*s. En la actualidad no se cuenta con una fuente de fundamentos teóricos y prácticos adecuada para el uso correcto de estos elementos, por lo cual se incurre en gastos innecesarios de dinero y tiempo.

La información incluida se basa en el documento de apoyo para los alumnos del Laboratorio de PIC. Esta información se basa en los temas menos conocidos por los estudiantes, por lo cual el investigador recolectó información pertinente. El documento incluye una parte teórica donde se le explica cómo seleccionar un PIC dependiendo de las necesidades de diseño y la manera correcta de conectar el PIC, dependiendo de su uso.

El primer capítulo de programación se enfoca en un lenguaje de alto nivel con el cual la mayoría de estudiantes está familiarizado. De esta manera, la comprensión y adaptación del conocimiento para el uso de un PIC es más fácil de procesar para el estudiante. Luego se pasa a un lenguaje de bajo nivel ASM. La comprensión de este lenguaje es más compleja. Para programar en ASM se usan las bases de conocimiento adquiridas en los ejemplos de lenguaje de alto nivel

También se comparan dos lenguajes de programación, uno de bajo nivel (ASM) y otro de alto nivel (BASIC) para que el estudiante identifique mejor las ventajas y desventajas de cada lenguaje para decidir cuál se adapta mejor a sus necesidades de diseño.

lenguaje y partiendo de allí pueda decidir cuál es el que se adapta mejor a sus necesidades de diseño.

OBJETIVOS

General

Recopilar información para facilitar el aprendizaje de los PIC (microcontroladores) y evitar la inversión innecesaria de tiempo y dinero en las pruebas de funcionamiento de circuitos y configuraciones de módulos para elaborar proyectos donde se necesite una interacción y autonomía más compleja.

Específicos

1. Explicar cómo optimizar los puertos de un PIC, la manera correcta de conectarlo, como usar los módulos más básicos, configurar los distintos tipos de reloj.

2. Utilizar dos lenguajes de programación BASIC y ASM para que el estudiante diferencie entre los distintos programas y seleccione de mejor manera el lenguaje de programación de acuerdo con sus necesidades.

3. Utilizar el programa *Pic Simulator Ide* ya que los módulos de simulación son de gran apoyo al momento de estar programando el PIC y no necesario tener un PIC físicamente para estar probando que el programa funciona correctamente, además la licencia de uso es muy barata en comparación de otros programas que tienen el mismo fin, el único impedimento que se tiene, si el estudiante se decide usar el programa en

su versión gratuita. Este programa solo se puede usar 30 veces en periodos de 2 horas antes de quedar deshabilitado su uso.

INTRODUCCIÓN

El presente trabajo de graduación ha sido desarrollado para ayudar al estudiante en el aprendizaje de la programación de PIC (Microcontroladores) ya que por ser un tema no muy explorado en la Escuela de Mecánica Eléctrica resulta complicado encontrar un punto de apoyo para ciertas dudas que se trataran de explicar en este trabajo.

Es una guía por los empleos y ejercicios cuya finalidad es el aprendizaje de cómo integrar los conocimientos adquiridos y solucionar un problema de la manera más eficiente. Este punto es muy importante en los microcontroladores dado que es un sistema cerrado a espacio de almacenamiento, velocidad de procesamiento, interfaces de conexión.

El material de apoyo, para trabajar los distintos ejemplos se encuentra en el apéndice, contiene información importante para la elaboración de un proyecto: tablas de conversión numérica, diagramas de circuitos, tablas de elementos de configuración, set de instrucciones de trabajo y su explicación respectiva.

1. ¿QUÉ ES UN MICROCONTROLADOR?

Es un dispositivo electrónico programable que se encuentra en la mayoría de los elementos utilizados en la vida cotidiana. por lo cual no se les confiere importancia. Están en las alarmas de carros, luces led, luces de adornos navideños, cafeteras, refrigeradoras, etc.

Es una pequeña computadora que tiene periféricos de entrada y salida de información, como también áreas de almacenaje de información volátil y no volátil. Todos estos elementos pueden ser programados por el usuario para un fin determinado.

En un principio los componentes se encontraban físicamente separados y el usuario debía integrarlos en una placa. En la actualidad todos los elementos del controlador se han podido incluir en un chip, que recibe el nombre de microcontrolador o PIC. PIC es acrónimo de *Peripheral Interface Controller* (controlador de interfaz periférico) En un sencillo pero completo computador contenido en el corazón (chip) de un circuito integrado.

Al encapsular todos los elementos en un chip se revolucionó la industria de los elementos electrónicos porque abarató costos en todos los sentidos. Un microcontrolador es un circuito integrado de alta escala de integración que incorpora la mayor parte de los elementos que configuran un controlador.

2. ¿QUÉ COMPONE UN MICROCONTROLADOR?

Un microcontrolador dispone de los siguientes componentes**:**

- Procesador o CPU (Unidad Central de Proceso).
- Memoria volátil RAM para Contener los datos.
- Memoria no volátil para almacenar la rutina de trabajo
- Periféricos de E/S para comunicarse con el exterior.
- Módulos especiales para el control de funciones (temporizadores, Puertas Serie y Paralelo, CAD: Conversores Analógico/Digital, CDA: Conversores Digital/Analógico, entre otros.).
- Generador de impulsos de reloj que sincronizan el funcionamiento de todo el sistema.

3. VENTAJAS EN EL USO DE UN MICROCONTROLADOR (PIC)

El usos de microcontrolador es ventajoso si se le compara con un circuito que contenga sus componentes (periféricos, memoria, CPU) por separado porque reduce los elementos de operación a un solo encapsulado, los puntos de falla, aumenta el control de procesamiento, la flexibilidad al realizar mejoras en el programa de ejecución, se reduce espacio físico de uso, cuando falla solo se reemplaza una pieza, los tiempos de procesamiento de datos entre memorias son menores, hay operaciones matemáticas que debemos usar cálculos con punto flotante.

Las nuevas tendencias de Internet de las Cosas *IoT* y domótica necesitan elementos versátiles en comunicación y procesamiento de información, que consuman poca de energía.

4. APLICACIONES DE LOS MICROCONTROLADORES

Continuamente más soluciones comerciales incorporan un microcontrolador para aumentar sus prestaciones, reducir su tamaño y coste, mejorar su fiabilidad y disminuir el consumo.

Los microcontroladores se usan en multitud de sistemas, de la vida cotidiana como pueden ser juguetes, drones, refrigeradoras, televisores, *mouses*, impresoras, *módems*, el sistema de arranque de nuestro carro, alarmas de carros, sistemas autónomos de vigilancia, etc. Una aplicación típica podría emplear varios microcontroladores para controlar pequeñas partes del sistema. Estos pequeños controladores podrían comunicarse entre ellos y con un procesador central, más potente, para compartir la información y coordinar sus acciones.

Este concepto de comunicación da origen a un sinfín de posibilidades, entre ellas *BIG DATA*, que se enfoca en la adquisición de la mayor cantidad de información para analizar tendencias de consumo, servicio o proceso, para buscar puntos de mejora en los sistemas.

Los microcontroladores son necesarios para adquirir datos para *BIG DATA* ya que, al ser versátiles en sus periféricos de comunicación, integran gran cantidad de sensores sin importar su manera de comunicación y medición. Al tener un microcontrolador en el sistema, se generan de ordenes de reacción proveniente del análisis de los datos obtenidos en tiempo real y con esto se logra los ajustes para tener en óptimas condiciones nuestros procesos.

5. ARQUITECTURA BÁSICA

Aunque inicialmente todos los microcontroladores adoptaron la arquitectura clásica de von Neumann, en el momento presente se impone la arquitectura Harvard. La arquitectura de von Neumann se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control). La arquitectura Harvard dispone de dos memorias independientes una, que contiene solo instrucciones y otra, solo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias.

6. UNIDAD CENTRAL DE PROCESO DEL PIC

Es el elemento más importante del microcontrolador y determina las principales características de hardware como *software*. Ejecuta las operaciones para el desarrollo del programa.

En la industria de los microprocesadores existen 3 sistemas de funcionalidad y de arquitectura, cada solución está enfocada a un campo en específico:

6.1. Cics

Computadores de juego de instrucciones complejo. Los dispositivos que trabajan con este sistema poseen más de 80 instrucciones, algunas son muy sofisticadas y potentes, por consiguiente, agilizan el procesamiento de información.

6.2. Risc

Computadores de juego de instrucciones reducido. Los dispositivos que trabajan con este set de instrucciones cuentan con un repertorio muy limitado y simple, generalmente, se ejecutan en un ciclo. La sencillez y rapidez de las instrucciones permiten optimizar el *hardware* y el *software* del procesador.

6.3. Sisc

Computadores de juego de instrucciones específico. Los dispositivos que trabajan con este set de instrucciones están destinados a aplicaciones muy concretas, el set de instrucciones es específico, las instrucciones se adaptan a las necesidades de la aplicación prevista.

7. MEMORIA

En los microcontroladores hay dos tipos de memoria: memoria de instrucciones y memoria de datos, ambas se encuentran integradas en el propio encapsulado. Una parte de memoria debe ser no volátil (*ROM*), y su función es contener el programa de instrucciones que gobierna la aplicación. Otra parte de memoria debe ser tipo volátil (*RAM*), y se destina a guardar las variables y los datos generados por la compilación del programa. Se debe tener en cuenta que no existen sistemas de almacenamiento masivo como disco duro o disquetes, el límite es la memoria contenida en el chip.

Como el microcontrolador tiene solo una tarea de trabajo la memoria *ROM* es muy limitada.

La *RAM* en estos dispositivos es de poca capacidad pues solo debe contener las variables y los cambios de información que se produzcan en el transcurso del programa. Por otra parte, como sólo existe un programa activo, no se requiere guardar una copia del mismo en la *RAM* pues se ejecuta directamente desde la *ROM*.

En la actualidad al referirse a la capacidad de almacenaje ser utilizamos términos en *Gigabytes* de memoria, en contraste en los PIC se trabajan capacidades de *ROM* comprendidas entre 512 *bytes* y 8 *kilobytes*, y de *RAM* comprendidas entre 20 y 512 *bytes*.

Según el tipo de memoria *ROM* que dispongan los microcontroladores, su aplicación y utilización es diferente. Se describen las cinco versiones de

memoria no volátil que se pueden encontrar en los microcontroladores del mercado.

7.1. Rom con máscara

Es una memoria no volátil de solo lectura cuyo contenido se graba durante la fabricación del chip. El elevado coste del diseño de la máscara sólo hace aconsejable el empleo de los microcontroladores con este tipo de memoria cuando se precisan cantidades superiores a varios miles de unidades.

7.2. OTP

El microcontrolador contiene una memoria no volátil de solo lectura "programable una sola vez" por el usuario. OTP (*One Time Programmable*). Es el usuario quien puede escribir el programa en el chip mediante un sencillo grabador controlado por un programa desde un PC.

7.3. EPROM

Los microcontroladores que disponen de memoria *EPROM* (*Erasable Programmable Read OnIy Memory*) pueden borrarse y grabarse muchas veces. La grabación se realiza, como en el caso de los OTP, con un grabador gobernado desde un PC. Si, posteriormente, se desea borrar el contenido, disponen de una ventana de cristal en su superficie por la que se somete a la *EPROM* a rayos ultravioleta durante varios minutos. Las cápsulas son de material cerámico y son más caros que los microcontroladores con memoria OTP que están hechos con material plástico.

7.4. EEPROM

Se trata de memorias de solo lectura, programables y borrables eléctricamente *EEPROM* (*Electrical Erasable Programmable Read Only Memory*). Tanto la programación como el borrado se realizan eléctricamente desde el propio grabador y bajo el control programado de un PC. Es muy cómoda y rápida la operación de grabado y la de borrado. No disponen de ventana de cristal en la superficie.

Los microcontroladores dotados de memoria *EEPROM* una vez instalados en el circuito, pueden grabarse y borrarse cuantas veces se quiera sin ser retirados de dicho circuito. Para ello se usan "grabadores en circuito" que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo.

El número de veces que puede grabarse y borrarse una memoria *EEPROM* es finito, por lo que no es recomendable una reprogramación continua. Son muy idóneos para la enseñanza y la Ingeniería de diseño.

7.5. FLASH

Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una *ROM* y una *RAM,* pero consume menos y es más pequeña. A diferencia de la *ROM*, la memoria *FLASH* es programable en el circuito. Es más rápida y de mayor densidad que la *EEPROM*. Las memorias *EEPROM* y *FLASH* son muy útiles al permitir que los microcontroladores que las incorporan puedan ser reprogramados "en circuito", es decir, sin tener que sacar el circuito integrado de la tarjeta. Así, un dispositivo con este tipo de memoria incorporado al control del motor de un automóvil

permite que pueda modificarse el programa durante la rutina de mantenimiento periódico, compensando los desgastes y otros factores como la compresión, la instalación de nuevas piezas, entre otros. La reprogramación del microcontrolador puede convertirse en una labor rutinaria dentro de la puesta a punto.

La memoria *RAM* es una memoria volátil, es decir, que se pierden los datos al desconectar el equipo, y se destina a guardar las variables y los datos. Los microcontroladores disponen de capacidades de *RAM* comprendidas entre 20 y 512 bytes.

8. RELOJ

Todos los microcontroladores disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema.

Generalmente, el circuito de reloj está incorporado en el microcontrolador y solo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Dichos componentes suelen consistir en un cristal de cuarzo junto a elementos pasivos o bien un resonador cerámico o una red R-C. También está la opción de osciladores internos los cuales son seleccionados su velocidad de trabajo por medio de comandos. También hay osciladores de baja potencia en donde no se necesita alta precisión y nuestras limitantes de potencia de trabajo son altas.

Hay que tomar en cuenta que un aumento en la velocidad de reloj repercute directamente en la disminución del tiempo de ejecución de las instrucciones, pero esto genera un incremento del consumo de energía.

9. RECURSOS ESPECIALES DE LOS MICROCONTROLADORES

Cada fabricante oferta numerosas versiones de una arquitectura básica de microcontrolador. En algunas amplía las capacidades de las memorias, en otras incorpora nuevos recursos, en otras reduce las prestaciones al mínimo para aplicaciones muy simples, entre otros. La labor del diseñador es encontrar el modelo mínimo que satisfaga todos los requerimientos de su aplicación. De esta forma, minimizará el coste, el hardware y el software.

Los principales recursos específicos que incorporan los microcontroladores son:

- Temporizadores o *Timers*.
- Perro guardián o *Watchdog*.
- Protección ante fallo de alimentación o *Brownout*.
- Estado de reposo o de bajo consumo.
- Conversor A/D.
- Conversor D/A.
- Comparador analógico.
- Modulador de ancho de pulsos o *PWM*.
- Puertas de E/S digitales.
- Puertas de comunicación.

9.1. Temporizadores o *Timers*

Se emplean para controlar periodos de tiempo (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores).

Para la medida de tiempos se carga un registro con el valor adecuado y a continuación dicho valor se va incrementando o decrementando al ritmo de los impulsos de reloj o algún múltiplo hasta que se desborde y llegue a 0, momento en el que se produce un aviso.

Cuando se desean contar acontecimientos que se materializan por cambios de nivel o flancos en alguna de las patitas del microcontrolador, el mencionado registro se va incrementando o decrementando al ritmo de dichos impulsos.

9.2. Perro guardián o *Watchdog*

Cuando el computador personal se bloquea por un fallo del *software* u otra causa, se pulsa el botón del *reset* y se reinicializa el sistema. Pero un microcontrolador funciona sin el control de un supervisor y de forma continuada las 24 horas del día. El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un *reset* automáticamente en el sistema.

Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al Perro guardián antes de que provoque el *reset*. Si falla el programa o se bloquea, no se refrescará al Perro guardián y, al completar su temporización, "ladrará y ladrará" hasta provocar el *reset*.

9.3. Protección ante fallo de alimentación o *Brownout*

Se trata de un circuito que resetea al microcontrolador cuando el voltaje de alimentación (VDD) es inferior a un voltaje mínimo. Mientras el voltaje de alimentación sea inferior al de *brownout* el dispositivo se mantiene reseteado, comenzando a funcionar normalmente cuando sobrepasa dicho valor.

9.4. Estado de reposo o de bajo consumo

Son abundantes las situaciones reales de trabajo cuando el microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, (factor clave en los aparatos portátiles), los microcontroladores disponen de una instrucción especial (*SLEEP* en los PIC), que le pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos. En dicho estado se detiene el reloj principal y se "congelan" sus circuitos asociados, quedando sumido en un profundo "sueño" el microcontrolador. Al activarse una interrupción ocasionada por el acontecimiento esperado, el microcontrolador se despierta y reanuda su trabajo.

9.5. Conversor A-D (CAD)

Los microcontroladores que incorporan un conversor A/D (Analógico/Digital) pueden procesar señales analógicas, abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD diversas señales analógicas desde las patitas del circuito integrado.

9.6. Comparador analógico

Algunos modelos de microcontroladores disponen internamente de un Amplificador Operacional que actúa como comparador entre una señal fija de referencia y otra variable que se aplica por una de las patitas de la cápsula. La salida del comparador proporciona un nivel lógico 1 o 0 según una señal sea mayor o menor que la otra.

También hay modelos de microcontroladores con un módulo de tensión de referencia que proporciona diversas tensiones de referencia aplicables en los comparadores.

9.7. Modulador de ancho de pulsos (PWM)

Son circuitos que proporcionan en su salida impulsos de anchura variable, que se ofrecen al exterior a través de las patitas del encapsulado.

9.8. Puertos de E/S digitales

Todos los microcontroladores destinan algunas de sus patitas a soportar líneas de E/S digitales. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertos.

Las líneas digitales de los Puertos pueden configurarse como entrada o como salida cargando un 1 o un 0 en el bit correspondiente de un registro destinado a su configuración.

9.9. Puertos de comunicación

Con objeto de dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas, buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos. Algunos modelos disponen de recursos que permiten directamente esta tarea, entre los que destacan:

- *UART*, adaptador de comunicación serie asíncrona.
- *USART*, adaptador de comunicación serie síncrona y asíncrona puerta paralela esclava para poder conectarse con los buses de otros microprocesadores.
- *USB* (Universal Serial Bus), que es un moderno bus serie para los PC.
- Bus *I2C*, que es un interfaz serie de dos hilos desarrollado por *Philips*.
- CAN (*Controller Area Network*), para permitir la adaptación con redes de conexionado multiplexado desarrollado conjuntamente por Bosch e Intel para el cableado de dispositivos en automóviles. En EE. UU. se usa el J185O.

10. CÓMO SE USAN LOS MICROCONTROLADORES EN LA INDUSTRIA

Existe una gran diversidad de microcontroladores. Quizá la clasificación más importante sea entre microcontroladores de 4, 8, 16 o 32 bits. Aunque las prestaciones de los microcontroladores de 16 y 32 bits son superiores a los de 4 y 8 bits, la realidad es que los microcontroladores de 8 bits dominan el mercado y los de 4 bits se resisten a desaparecer. La razón de esta tendencia es que los microcontroladores de 4 y 8 bits son apropiados para la gran mayoría de las aplicaciones, lo que hace absurdo emplear micros más potentes y consecuentemente más caros.

La distribución de las ventas según su aplicación se da en una tercera parte en las aplicaciones relacionadas con los computadores y sus periféricos. Una cuarta parte se utiliza en las aplicaciones de consumo doméstico y el resto en aplicaciones de telecomunicaciones. Como se comentó anteriormente una de las últimas tendencias es la adquisición de información de forma masiva *BIGDATA*, esto viene a dar más importancia a los PIC, ya que el problema hoy por hoy de esta tendencia es como integrar a la red de datos dispositivos que tiene medio de conexión.

11. REQUISITOS QUE SE DEBEN EVALUAR AL SELECCIONAR UN MICROCONTROLADOR PARA UNA APLICACIÓN DETERMINADA

11.1. Procesamiento de datos

Puede ser necesario que el microcontrolador realice cálculos críticos en un tiempo limitado. En ese caso, se debe seleccionar un dispositivo suficientemente rápido para ello. Por otro lado, habrá que tener en cuenta la precisión de los datos que se manejarán: si no es suficiente con un microcontrolador de 8 *bits*, puede ser necesario acudir a microcontroladores de 16 o 32 *bits*, o incluso a hardware de coma flotante. Una alternativa más barata y quizá suficiente es usar librerías para manejar los datos de alta precisión.

11.2. Entrada salida

Para determinar las necesidades de entrada/salida del sistema es conveniente dibujar un diagrama de bloques, de tal forma que sea sencillo identificar la cantidad y tipo de señales por controlar. Una vez realizado este análisis puede ser necesario añadir periféricos hardware externos o cambiar a otro microcontrolador más adecuado a ese sistema.

11.3. Consumo

Algunos productos que incorporan microcontroladores están alimentados con baterías y su funcionamiento puede ser tan vital como activar una alarma antirrobo. Lo más conveniente en un caso como éste puede ser que el

microcontrolador esté en estado de bajo consumo pero que despierte ante la activación de una señal (una interrupción) y ejecute el programa adecuado para procesarla. Un ejemplo de estos son los *Zigbee*, estos PIC están diseñados para la adquisición de información, pero su bajo consumo lo hace uno de los elementos más recomendados.

11.4. Memoria

Para detectar las necesidades de memoria de la aplicación debe separarse en memoria volátil (*RAM*), memoria no volátil (*ROM*, *EPROM, entre otros*) y memoria no volátil modificable (*EEPROM*). Este último tipo de memoria puede ser útil para incluir información específica de la aplicación como un número de serie o parámetros de calibración.

En cuanto a la cantidad de memoria necesaria puede ser imprescindible realizar una versión preliminar, aunque sea en pseudo-código, de la aplicación y a partir de ella hacer una estimación de cuánta memoria volátil y no volátil es necesaria y si es conveniente disponer de memoria no volátil modificable.

11.5. Ancho de palabra

El criterio de diseño debe ser seleccionar el microcontrolador de menor ancho de palabra que satisfaga los requerimientos de la aplicación. Usar un microcontrolador de 4 bits supondrá una reducción en los costes importante, mientras que uno de 8 bits puede ser el más adecuado si el ancho de los datos es de un byte. Los microcontroladores de 16 y 32 bits, debido a su elevado coste, deben reservarse para aplicaciones que requieran sus altas prestaciones (Entrada/Salida potente o espacio de direccionamiento muy elevado).

11.6. Diseño de la placa

La selección de un microcontrolador concreto condicionará el diseño de la placa de circuitos. Debe tenerse en cuenta que quizá usar un microcontrolador barato encarezca el resto de componentes del diseño.

12. HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES

Uno de los factores que más importancia tiene al seleccionar un microcontrolador entre todos los demás es el soporte tanto software como hardware de que dispone. Un buen conjunto de herramientas de desarrollo puede ser decisivo en la elección, ya que pueden suponer una ayuda inestimable en el desarrollo del proyecto.

Las principales herramientas de ayuda al desarrollo de sistemas basados en microcontroladores son las siguientes:

12.1. Ensamblador

La programación en lenguaje ensamblador puede resultar ardua para el principiante, pero permite desarrollar programas muy eficientes, ya que otorga al programador el dominio absoluto del sistema. Los fabricantes generalmente, proporcionan el programa ensamblador de forma gratuita y en cualquier caso siempre se puede encontrar una versión gratuita para los microcontroladores más populares.

12.2. Compilador

La programación en un lenguaje de alto nivel (como en *C* o el *Basic*) disminuye el tiempo de desarrollo de un producto. No obstante, si no se programa con cuidado, el código resultante puede ser mucho más ineficiente que el programado en ensamblador. Las versiones más potentes suelen ser

muy caras, aunque para los microcontroladores más populares pueden encontrarse versiones demo limitadas e incluso compiladores gratuitos.

Debido a que los microcontroladores van a controlar dispositivos físicos, los desarrolladores necesitan herramientas para comprobar el buen funcionamiento del microcontrolador cuando es conectado al resto de circuitos.

12.3. Simulador

Son programas que se ejecutan en una computadora y su función es compilar los programas realizados para el microcontrolador. Los simuladores tienen un control absoluto sobre la ejecución de un programa, son ideales para la depuración de los mismos. Su gran inconveniente es que es difícil simular la entrada y salida de datos del microcontrolador. Tampoco cuentan con los posibles ruidos en las entradas, pero, al menos, permiten el paso físico de la implementación de un modo más seguro y menos costoso, puesto que se ahorra en grabaciones de chips para la prueba en vivo.

12.4. Placas de evaluación

Se trata de pequeños sistemas con un microcontrolador ya montado y que generalmente, se conectan a un PC desde el que se cargan los programas que se ejecutan en el microcontrolador. Las placas suelen incluir visualizadores LCD, teclados, leds, fácil acceso a los pines de E/S, entre otros. El sistema operativo de la placa recibe el nombre de programa monitor. El programa monitor de algunas placas de evaluación, aparte de cargar programas y datos en la memoria del microcontrolador, puede permitir en cualquier momento realizar ejecución paso a paso, monitorizar el estado del microcontrolador o modificar los valores almacenados los registros o en la memoria.

12.5. Emuladores de circuitos

Se trata de un instrumento que se coloca entre el PC anfitrión y el zócalo de la tarjeta de circuito impreso donde se alojará el microcontrolador definitivo. El programa es ejecutado desde el PC, pero para la tarjeta de aplicación es como si lo hiciese el mismo microcontrolador que luego irá en el zócalo. Presenta en pantalla toda la información tal y como luego sucederá cuando se coloque la cápsula.

Para transferir el código programado hacia el microprocesador PIC, Microchip ha implementado el ICSP (*In Circuit Serial Programming*, programación serial incorporado) o LVP (*Low Voltaje Programming*, programación a bajo voltaje), para programar el PIC directamente al circuito de destino.

La programación ICSP usa pines según las características propias del microprocesador, RB6 y RB7 ó GP0 Y GP1 o el RA0 y RA1, como reloj y datos. Para activar la rutina de programación se aplica un voltaje de 13 voltios al pin MCLR.

Existe una diversidad de sistemas para descargar y depurar en línea desde los más simples que dejan al software los detalles de comunicación, a los más complejos que pueden verificar el dispositivo a diversas tensiones de alimentación e implementan en hardware casi todas las funciones.

A continuación, se enlista algunos programadores:

- *PICStart* Plus (puerto serie y USB)
- *Promate* II (puerto serie)
- MPLAB PM3 (puerto serie y USB)
- ICD2 (puerto serie y USB)
- ICD3 (USB)
- PICKit 1(USB)
- IC-PROG 1.06
- PICAT 1.25 (PUERTO USB2.0)
- *WinPic* 800 (puerto paralelo)
- PICKIT2
- PICKIT3
- Terusb
- Eclipse (PIC y AVR USB)
- *MasterPRO*

Depuradores integrados

- Proteus –ISIS
- ICE2000 (Puerto paralelo, convertidor a USB)
- ICE4000(USB)
- PIC EMU
- ISEC
- PIC CDlite
- PIC SIMULATOR

13. EVOLUCIÓN DE MICROPROCESADORES (PIC)

La evolución de tecnología en los microprocesadores pic no se ha quedado atrás y existen diferentes variantes según las necesidades.

13.1. PICs wireless

El microcontrolador rfPIC integra todas las prestaciones del *PICmicro* de microchip con la capacidad de comunicación wireless *UHF* para aplicaciones RF de baja potencia. Estos dispositivos ofrecen un diseño muy comprimido para ajustarse a los cada vez más demandados requerimientos de miniaturización en aparatos electrónicos. Aun así, no parecen tener mucha salida en el mercado.

13.2. PICs para procesado de señal (dsPICs)

Los *dsPICs* son el penúltimo lanzamiento de Microchip, su producción a gran escala inició a finales de 2004. Son los primeros PICs con bus de datos inherente de 16 bits. Incorporan todas las posibilidades de los anteriores PICs y añaden varias operaciones de DSP implementadas en hardware, como multiplicación con suma de acumulador (*multiply-accumulate*, o *MAC*), *barrel shifting*, *bit reversion* o multiplicación 16x16 bits.

13.3. PICs de 32 bits (PIC32)

Microchip Technology lanzó en noviembre de 2007 los nuevos microcontroladores de 32 bits con una velocidad de procesamiento de 1.5

DMIPS/MHz con capacidad HOST USB. Estos MCUs permiten un elevado procesamiento de información, con un núcleo de procesador de tipo M4K.

A continuación, se enlistarán algunos pic y para que se usan normalmente.

- PIC12C508/509 (encapsulamiento reducido de 8 pines, oscilador interno, popular en pequeños diseños como el *iPod remote*).
- PIC16F84 (Considerado obsoleto, pero imposible de descartar y muy popular)
- PIC16F84A (Buena actualización del anterior, algunas versiones funcionan a 20 MHz, compatible 1:1)
- PIC16F628A (Es la opción típica para iniciar una migración o actualización de diseños antiguos hechos con el PIC16F84A. Posee puerto serial, módulos de comparación análoga, *PWM*, módulo *CCP*, rango de operación de voltaje aumentado, entre otras)
- PIC16F88 (Nuevo sustituto del PIC16F84A con más memoria, oscilador interno, *PWM*, etc. que podría convertirse en popular como su hermana).
- La subfamilia PIC16F87X y PIC16F87XA (los hermanos mayores del PIC16F84 y PIC16F84A, con cantidad de mejoras incluidas en hardware.
- PIC16F886/887 (Nuevo sustituto del 16F876A y 16F877A con la diferencia que el nuevo ya se incluye oscilador interno).
- PIC16F193x (Nueva gama media de PIC optimizado y con mucha *RAM*, ahora con 49 instrucciones por primera vez frente a las 35 de toda la vida).
- PIC18F2455 y similares con puerto USB 2.0
- PIC18F2550 manejo de puertos USB 2.0 y muy versátil.
- dsPIC30F3011 (Ideales para control electrónico de motores eléctricos de inducción, control sobre audio, etc.).

- PIC32 (Nueva gama de PIC de 32 bits, los más modernos ya compatible con USB 2.0).

Un microprocesador necesita los valores digitales 1 digital y 0 digital para trabajar de manera correcta, dichos valores son. Es importante tomar en cuenta estos datos al diseñar un proyecto, ya que en la familia de los micro procesadores existen varias familias en las cuales los valores de voltaje con los que trabajan son distintos.

En esta investigación se trabajarán los microprocesadores que su funcionamiento es del rango de 0 a 5 voltios.

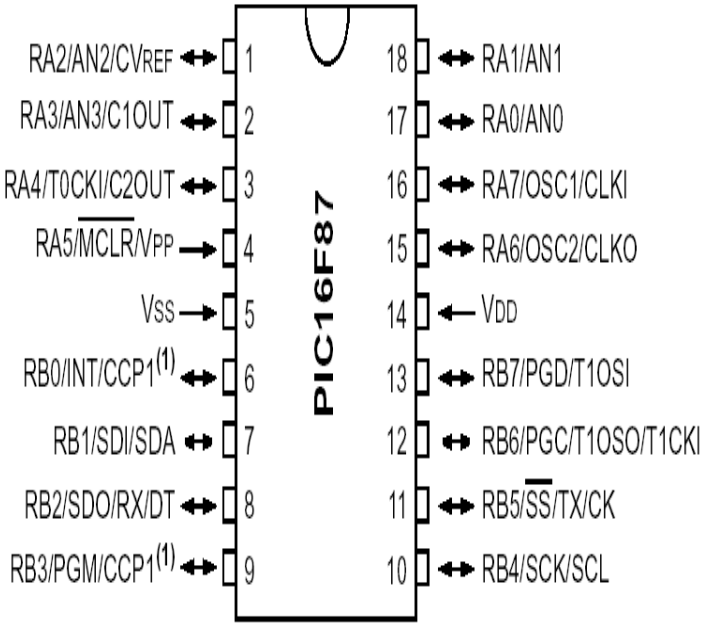
Un puerto se compone de 8 posiciones dando como resultado un valor máximo en decimal de 255 posibles valores o su equivalente en otros sistemas de numeración. Cada posición o pin de un microprocesador puede ser programada independientemente como dispositivo de entrada o salida según sea la necesidad.

Es importante seleccionar el microprocesador más adecuado a las necesidades del proyecto, como los siguientes parámetros.

- Capacidad de memoria
- Tipo de memoria
- Cantidad de puertos
- Puertos con especificaciones únicas en el dispositivo
- Tipo de alimentación
- Tipo de reloj de trabajo
- Espacio físico para el microprocesador

Estos parámetros son importantes ya que se vinculan al precio del microprocesador. Si las prestaciones del microprocesador no se adecuan al proyecto aumentan los costos y si se compra uno que excede las necesidades del proyecto se pierde dinero.

Figura 1. **Pic 16F87**



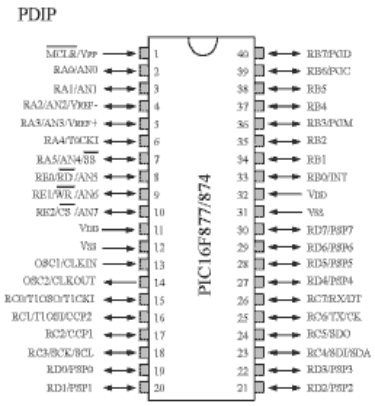
Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf.
Consulta: 11 de noviembre 2017.

Tabla I. **Características de pic 16F87**

Parameter Name	Value	Parameter Name	Value
Program Memory Type	Flash	Capture/Compare/PWM Peripherals	1 CCP
Program Memory (KB)	7	Timers	x 8-bit, 1 x 16-bit
CPU Speed (MIPS)	5	Comparators	2
RAM Bytes	368	Temperature Range (C)	40 to 125
Data EEPROM (bytes)	256	Operating Voltage Range (V)	to 5.5
Digital Communication Peripherals	1-/E/USART, 1-SSP(SPI/I2C)	Pin Count	8

Fuente: elaboración propia.

Figura 2. **Pic 16F877A**



Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf.
Consulta: 8 octubre 2017.

Tabla II. **Características de pic 16F877A**

Parameter Name	Value	Parameter Name	Value
Program Memory Type	Flash	Capture/Compare/PWM Peripherals	2 CCP
Program Memory (KB)	14	Timers	2 x 8-.6-bit
CPU Speed (MIPS)	5	ADC	8 ch,
RAM Bytes	368	Temperature Range (C)	-40 to
Data EEPROM (bytes)	256	Operating Voltage Range (V)	2 to
Digital Communication Peripherals	1-A/E/USART, 1-MSSP (SPI/I2C)	Pin Count	40

Fuente: elaboración propia.

Si en el proyecto se deben usar dos puertos, uno con entrada analógica (ADC) y el otro como puerto de salida digital, se pueden usar las opciones anteriores tomando en cuenta que el 16F87 es un dispositivo de dos puertos. Como tiene oscilador interno no es necesario conectarle uno, en cambio el 16F877A, es un dispositivo que tiene 4 puertos de 8 Pines y 1 un puerto de 3 pines. El costo de cada 16F87 es de 3.50$ y el del 16F877A 7.45$

14. CÓMO IDENTIFICAR LAS CARACTERÍSTICAS DE UN PIC A PARTIR DE LA INFORMACIÓN IMPRESA EN SU ENCAPSULADO

Es importante saber el significado de la nomenclatura que poseen los Microprocesadores. Esta se separa en 6 partes:

PIC 16 F 877 A – I / P

14.1. Familia

Existen 3 familias en los microprocesadores, identificadas como de Bajo, Medio y Alto Rendimiento, Cada familia es identificada con un par de dígitos:

- Los dispositivos de la familia de bajo nivel son aquellos que poseen un nivel de proceso, almacenaje, desempeño bajo, carecen de puertos especializados. Se identifica con el digito 12.

- Los dispositivos de la familia de medio nivel son aquellos que presentan un performance medio, ya poseen puertos uno o varios puertos especializados en un mismo dispositivo. Se identifican con el digito 16.

- Los dispositivos de la familia de nivel alto son aquellos que presentan un alto desempeño en procesamiento, puertos especializados para actividades más robustas. Se identifican con el digito 18.

14.2. Tipo de memoria

Este parámetro nos indica que tipo de memoria posee el dispositivo, existen principalmente en dos tipos de memoria. Los de memoria Flash y los de memoria *Eprom*.

- Los de memoria *Flash* son aquellos dispositivos que soportan el ciclo de borrado y almacenamiento del programa en una cantidad determinada por el fabricante en 100,000 ciclos. Esto significa que se pueden realizar pruebas con el programa para ver su desempeño en el dispositivo. Se identifican con la letra F.

- Los de memoria *Eprom* o también llamados OTP (*One Time Programmable*) son aquellos que únicamente se puede descargar una vez el programa al dispositivo. Al concluir la descarga no se puede realizar ninguna alteración al programa ya que queda grabado sin la posibilidad de ser borrado. Se identifican con la letra C.

14.3. Registro

El registro es donde se observa si el integrado a tenido alguna actualización ya sea de software o de hardware por lo general un registro consta de 3 números. El cuarto número indica qué versión de integrado es 4550 4551 quiere decir que el integrado 455 a presentado algún tipo de actualización, pero en su estructura es la misma se puede sustituir el integrado 455 por un 4550, 4550, 455X.

14.4. Comparador analógico

La letra A se asocia que el integrado cuenta con registros especiales que permiten realizar una comparación de voltajes analógicos dentro del mismo microprocesador.

14.5. Rango de temperatura de trabajo

Son dos tipos de temperatura

Figura 3. **Rango de temperatura**

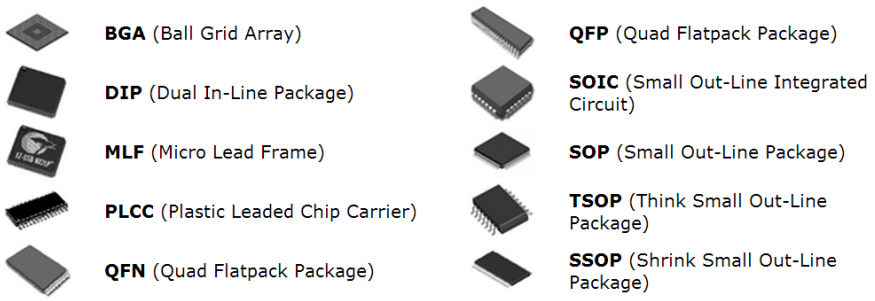
I = -40°C to +85°C	Temp. Industrial
E = -40°C to +125°C	Temp. Extendida

Fuente: elaboración propia.

14.6. Tipo de encapsulado

El tipo de encapsulado es muy importante porque indica qué forma y cantidad de pines tiene el dispositivo. Si se compra en línea y no supiéramos que escoger esta opción adecuadamente podemos echar a perder la comprar.

Figura 4. **Tipos de encapsulado**



Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf

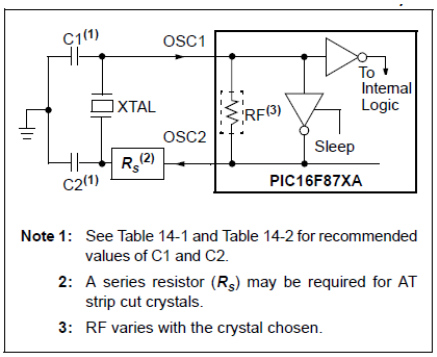
Consulta: 8 octubre 2017.

15. CÓMO CONFIGURAR EL RELOJ DEL MICROPROCESADOR

Las maneras de configurar el oscilador de un microprocesador dependerán de las particularidades del microprocesador. Algunos microprocesadores tienen 4 formas de configurar su oscilador, otros tienen 12 distintas maneras. A continuación, se explican 3 que son los básicos.

- Oscilador sobre sostenido
- Oscilador resistencia / capacitor
- Oscilador externo

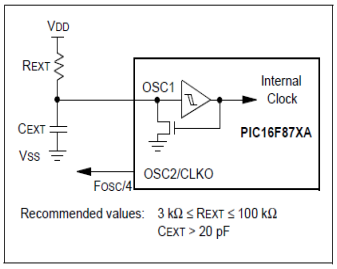
Figura 5. **Oscilador sobre sostenido**



Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf

Consulta: 8 octubre 2017.

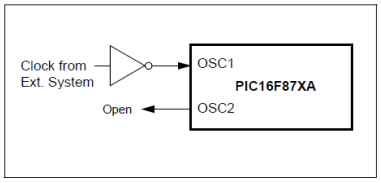
Figura 6. **Oscilador resistencia // capacitor**



Fuente http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf

Consulta: 8 octubre 2017.

Figura 7. **Oscilador externo**



Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf

Consulta: 8 octubre 2017.

15.1. Oscilador resistencia y capacitor

Esta configuración es externa al PIC y es una alternativa cuando no se tiene a la mano un oscilador de cristal, hay que tener en cuenta que este tipo de oscilador presenta inestabilidad.

15.2. Oscilador Interno

La ventaja de configurar un microprocesador con oscilador interno es que no se utiliza pin del microprocesador para ese uso, al igual que el oscilador de resistencia y capacitor, también presenta mucha variación de frecuencia. Esta opción es muy útil en la mayoría de los casos donde no se necesita exactitud.

15.3. Oscilador sobre sostenido

Este tipo de oscilador es externo y está compuesto por un oscilador de cristal, el cual basa su funcionamiento en el fenómeno piezoeléctrico y dos capacitores que se colocan entre el oscilador de cristal y el PIC.

Cada uno de esto tres tipos de osciladores básicos, tiene sus funciones especiales dependiendo del tipo de microprocesador.

Tabla III. **Configuración de osciladores**

Ranges Tested:			
Mode	**Freq.**	**OSC1**	**OSC2**
XT	455 kHz 2.0 MHz 4.0 MHz	68-100 pF 15-68 pF 15-68 pF	68-100 pF 15-68 pF 15-68 pF
HS	8.0 MHz 16.0 MHz	10-68 pF 10-22 pF	10-68 pF 10-22 pF
These values are for design guidance only. See notes following Table 14-2.			
Resonators Used:			
2.0 MHz	Murata Erie CSA2.00MG	± 0.5%	
4.0 MHz	Murata Erie CSA4.00MG	± 0.5%	
8.0 MHz	Murata Erie CSA8.00MT	± 0.5%	
16.0 MHz	Murata Erie CSA16.00MX	± 0.5%	
All resonators used did not have built-in capacitors.			

Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf

Consulta: 8 octubre 2017.

16. ARQUITECTURA INTERNA DE MICROPROCESADOR

Saber el tipo de arquitectura permite entender de mejor manera el procesamiento interno de los datos entre los registros de uso específico y los registros generales, también es importante tener una idea básica del funcionamiento interno, y el uso según la etiqueta que está en cada uno de los pines.

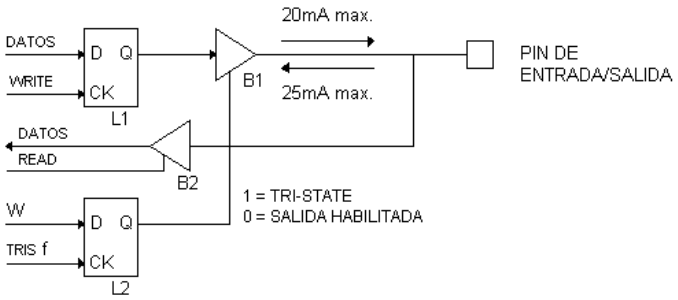
16.1. Capacidad de corriente en un pin

La máxima capacidad de corriente de cada uno de los pines de los puertos en modo sumidero (*sink*) es de 25 *mA* y en modo fuente (*source*) es de 20 *mA*.

16.2. Circuito equivalente de un pin I/O

El *latch* L1 corresponde a un bit del registro de datos del puerto, mientras que L2 es un bit del registro de control de *tristate* del mismo. B1 es el *buffer tristate* de salida que tiene capacidad de entregar 20 mA y drenar 25 mA. B1 es controlado por L2. Si L2 tiene cargado un “1”, B1 se encuentra en *tri-state*, es decir con la salida desconectada (en alta impedancia), y el puerto puede ser usado como entrada. Si L2 tiene cargado un “0”, la salida de B1 está conectada (baja impedancia) y el puerto está en modo de salida. B2 es el buffer de entrada, es decir el que pone los datos en el bus interno del microcontrolador cuando se lee el registro de datos del puerto. Puede verse que el dato leído es directamente el estado del pin de entrada.

Figura 8. **Circuito equivalente de un pin interno**



Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf

Consulta: 8 octubre 2017.

17. NOMENCLATURA DE PINES

La nomenclatura de pines indica, físicamente, dónde empieza y termina un puerto, cuántos pines los compone y sus propiedades. En algunos casos los PIC´s cuentan con un comparador analógico, el problema es saber identificar en qué puerto está la característica y en qué pines se debe colocar el oscilador. Estas y otras propiedades se identifican si se sabe la nomenclatura de los pines de un PIC.

Para esta explicación se utilizará el PIC 16F877A ya que cuenta con variedad de módulos y esto nos permite tener un mejor escenario de explicación. Se tabuló la información en dos columnas, la primera contiene el nemotécnico que posee el pin y la segunda columna, una pequeña explicación.

18. SET DE INSTRUCCIONES

Es el conjunto de instrucciones que una unidad central de proceso puede entender y ejecutar. Hay varias formas de clasificar el set de instrucciones, pero básicamente, se separan en:

18.1. Instrucciones orientadas a byte

Son las instrucciones que afectan o consultan el contenido de un registro por completo.

18.2. Instrucciones orientadas a bit

Son las instrucciones que afectan o consultan el contenido de un bit únicamente por ejemplo la consulta de una bandera de desborde.

18.3. Instrucciones orientadas a constantes y control

Son las instrucciones que se utiliza para hacer una llamada a un registro especial o al salto a una etiqueta dentro del programa.

La siguiente nomenclatura es básica para entender el funcionamiento de cada instrucción:

- f: Registro al que afecta la instrucción.
- W: Acumulador (*Working register*).
- b: Número de bit (hay instrucciones que afectan a un solo bit).

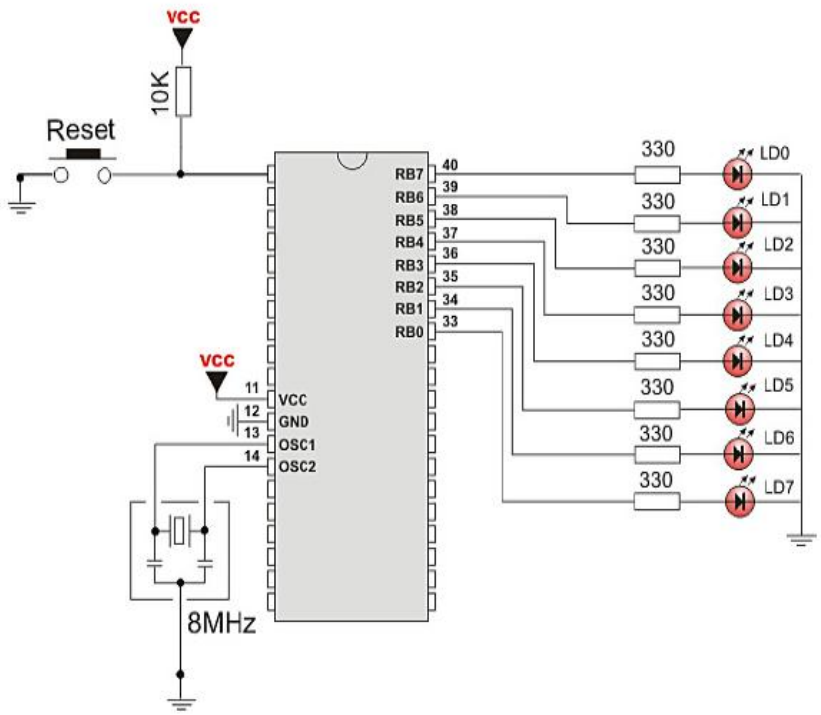
- k: constante (un número).
- d: selección de destino del resultado de la instrucción, puede ser "0" o "1", si es "0" el resultado se guarda en el acumulador (W) y si es "1" se guarda en el registro f al que afecta la instrucción.

El set de instrucciones se encuentra en los anexos

19. CÓMO CONECTAR POR PRIMERA VEZ UN PIC

A continuación, se muestra una imagen básica de cómo se debe conectar un PIC y el mismo circuito es el que estaremos utilizando para explicar los lenguajes de programación.

Figura 9. **Cómo conectar un pic mi primer circuito**



Fuente: elaboración propia.

19.1. Circuito de reset

El circuito conectado al Pin 1 (MCLR= *Master Clear Reset*) sirve para reiniciar el microprocesador este funciona cuando el valor lógico en la entrada es igual a un 0 lógico por consiguiente hay que ponerle una resistencia de desfogue conectada a un valor lógico de 1 para que el pic funcione. Esta configuración de 1 y 0 lógicos se debe a que el pin está negado. Por eso en las hojas de especificaciones del pic aparece este pin con una línea arriba de las letras. Al oprimir el botón generamos un 0 lógico y esto provoca que el microprocesador se reinicie.

19.2. Circuito de energización

Los pines 11 y 32 se deben conectar 5 voltios, los pines 12 y 31 se debe conectar 0 voltios. Con esto se polariza el microprocesador, pero surgen varias dudas. Qué pasaría si se energizan únicamente los pines 11 y 12 o si por el contrario se energiza el 32 y 31 únicamente. Básicamente la respuesta es la misma para todas las posibles variantes.

Por seguridad del sistema siempre deben estar polarizadas ambas fuentes de alimentación. De esta manera, la corriente de alimentación se distribuye entre ambas entradas de energía.

19.3. Circuito de oscilación

En estos pines se debe colocar el oscilador que se trabajará. En la imagen de ejemplo se usa un oscilador sobre sostenido.

19.4. Circuito de presentación

Este circuito está compuesto de un conjunto de 8 led y sus respectivas resistencias de protección que servirán para visualizar los resultados de los programas.

20. PROGRAMAS REALIZADOS EN LENGUAJE BASIC

A continuación, se desarrollarán varios programas en lenguaje BASIC siglas en ingles de *Beginner's All-purpose Symbolic Instruction Code*. Este lenguaje de programación es básicamente un programa desarrollado para facilitar el aprendizaje y programación de computadoras a estudiantes. Sus características especiales son el fácil uso para principiantes, ser un programa amigable al usuario, no necesita que el usuario sepa de hardware donde funcionara el programa.

El lenguaje Basic es un programa de alto nivel donde el usuario no necesita de conocimiento de complejo de programación. En la actualidad, se enseña en la mayoría de colegios de educación media. Esta es una ventaja para usarlo en los microprocesadores ya que su lenguaje se aprendido previamente y solo se debe aprender la adaptación hacia el mundo de los microprocesadores.

El lenguaje Basic se rige por medio de comandos y sintaxis que son mínimas y fáciles de. En este trabajo de graduación se darán a conocer los esenciales para trabajar con este.

Como en cualquier lenguaje de programación se debe conocer la sintaxis de los comandos por utilizar. Una de las propiedades del *PIC Simulator Ide*, al escribir alguna palabra reservada del programa, el sistema la reconoce y genera que un cambio de color en dicha palabra, esta propiedad también pasa con los registros, bancos de memoria, puertos del PIC.

Algo muy importante que debemos entender en la programación de un PIC indistintamente el lenguaje de programación es que el programa debe ser cíclicos, de lo contrario solo una vez realizara el programa. Para realizar ciclos dentro del programa podemos utilizar 3 instrucciones de programación que son básicos en cualquier lenguaje de programación: *FOR*, *WHILE* y *GOTO*. También tenemos las instrucciones comparativas *IF* y *CASE*

21. INSTRUCCIÓN FOR

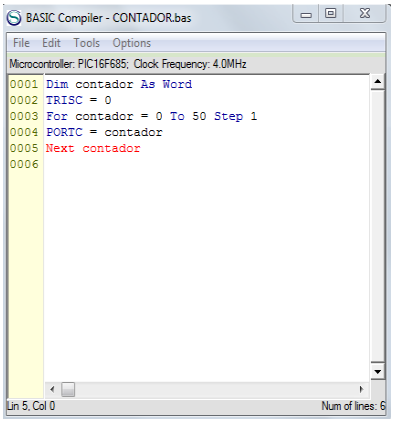
A continuación, se muestran dos opciones de programación de una instrucción FOR. En el programa A el ciclo For debe contar de manera ascendente y en el programa B, de manera descendente. La diferencia es el valor inicial y el valor de paso que utiliza.

Línea 1	Se declara una variable tipo Word que servirá de almacenadora de información.
Línea 2	Se declara el puerto C como puerto de salida.
Línea 3	Se inicia el ciclo FOR donde la variable almacenadora obtendrá el valor inicial y el valor de paso.
Línea 4	El valor numérico de la variable almacenadora es mostrado en el puerto que fue configurado como puerto de salida.
Línea 5	Fin del ciclo FOR.

En ambos ejemplos al finalizar el ciclo *For* el programa se queda esperando la siguiente instrucción. En este caso sería la línea.6 pero como no hay más instrucciones o condicionantes, entonces el programa se queda trabajando hacia el infinito sin hacer nada.

También se observa cómo las palabras reservadas están en un color azul, Cuando el simulador trabaja la línea que está compilando cambia a color rojo.

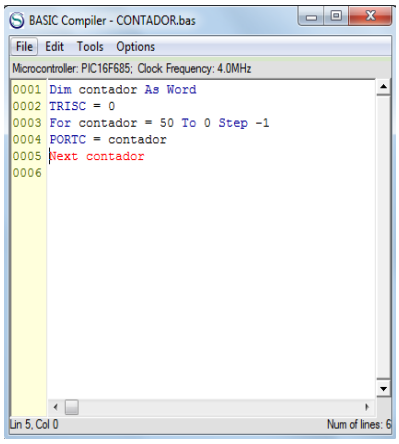
Figura 10. **Ciclo ascendente**



Fuente: elaboración propia.

Programa A

Figura 11. **Ciclo descendente**



Fuente: elaboración propia.

Programa B

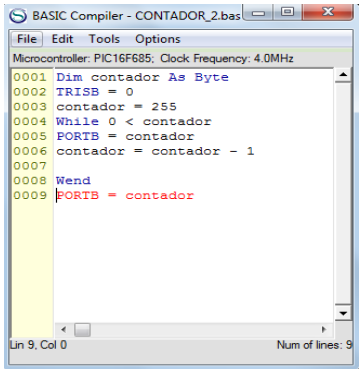
22. INSTRUCCIÓN WHILE

La instrucción While básicamente se ejecuta mientras su condicionante es verdadera de lo contrario se sale del ciclo.

Se declara una Variable tipo *Byte* que servirá de almacenadora de información.

Línea 1	Se declara una variable tipo *Byte* que servirá de almacenadora de información.
Línea 2	Se declara el puerto B como puerto de salida.
Línea 3	Se otorga un valor especifico a la almacenadora.
Línea 4	Se inicia el ciclo WHILE y se condiciona que se ejecute el ciclo mientras el valor de la variable es mayor a 0.
Línea 5	El valor numérico de la variable almacenadora se muestra en el puerto que fue configurado como puerto de salida.
Línea 6	El valor numérico de la variable almacenadora es disminuido en uno y asignado en nuevo valor a la misma variable.
Línea 8	Fin del ciclo WHILE.
Línea 9	Se muestra el valor final de la variable de almacenamiento en el puerto B

Figura 12. **Ciclo while**



Fuente: elaboración propia.

23. ¿CUÁNDO SE DEBE USAR LA INSTRUCCIÓN FOR O WHILE?

Cuando se necesita que el ciclo se ejecute por lo menos una vez se utiliza la instrucción FOR, y cuando se necesita hacer una comparación antes de un ciclo es mejor usar una instrucción WHILE.

Es importante seleccionar adecuadamente la instrucción correcta ya que se utiliza PIC que tienen poca capacidad de almacenamiento y si nuestro programa no es óptimo pueden enfrentar problemas de almacenamiento.

INSTRUCCIÓN GOTO

Esta instrucción permite crear un salto a una posición dentro del programa, ya sea para crear un *loop* infinito o una subrutina condicionada a un estado.

INSTRUCCIÓN IF

Esta instrucción hace comparaciones en relación con un valor definido y a partir de allí se toma una decisión. La instrucción *IF* la podemos combinar con las condicionantes matemáticas (>,<,>>,<<,!=,==) o anidarlas con las operaciones booleanas (*AND,OR,NOT*).

En este caso que necesitamos compara varios estados relacionados con diferentes referencias es aconsejable utilizar la instrucción *IF*.

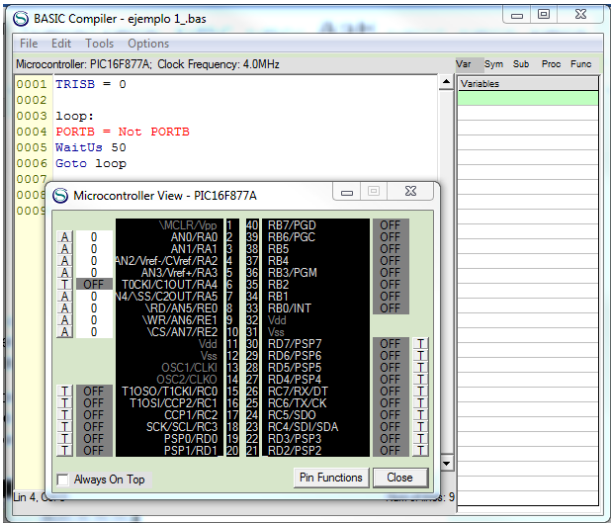
INSTRUCCIÓN CASE

En el caso que se necesita comparar varios valores definidos sobre una variable es mejor utilizar la instrucción *case*, el único inconveniente es que no podemos hacer condicionantes con otra variable a menos que se use un *if* dentro de una de las opciones del *case*, esto representaría más espacio de memoria, por ello, se debe analizar el escenario para saber qué analizar.

24. CONFIGURACIÓN DE PUERTO DE SALIDA

Este es el primer programa en un PIC. El objetivo de este ejemplo es hacer centellar varios leds, mediante la configuración de un puerto del PIC como puerto de salida de información. Para observar el funcionamiento se conectan leds a los pines programados.

Figura 13. **Primer programa**



Fuente: elaboración propia.

A continuación, se explicará detalladamente cada línea de programación

Línea 1 El comando TRISB hace referencia al registro TRIS y se utiliza para determinar el puerto B como puerto de entrada. Al colocar 0 se debe recordar que podemos colocar en tres formatos decimal (0), hexadecimal (0X00) y binario (%00000000). Es una etiqueta que servirá de retorno, las etiquetas se utilizan para dar un orden al programa y también como puntos de referencia cuando el programa se vuelve muy extenso.

Línea 2 Con el comando PORTB =0 indicamos al PIC que ponga sus pines del puerto B en 0 lógico como salida, también se puede utilizar cualquiera de los formatos *DEC*, *HEX* y *BIN*.

Línea 3 *WaitUS* seguido de un número es un *Timer* que demorara X microsegundos. Este comando también se puede cambiar por *WaitMs.* La diferencia es que el tiempo de demora ahora será medido en milisegundos.

Línea 4 El comando *NOT* niega el valor que tiene la variable, si es 1 lo convierte en 0 y si es 0 lo convierte en 1.

Línea 5 Comando de retardo *WaitUs* en microsegundos, *WaitMs* en milisegundos.

Línea 6 Con el comando *Goto* se indica que vaya a un determinado punto del programa puede ser una línea específica o como en este caso una etiqueta.

Al correr el programa se podrá observar que todos los pines del Puerto B empezaron a centellar en conjunto en un ciclo infinito. La única manera de parar dicho ciclo será des energizando el microprocesador.

El tiempo entre encendido y apagado lo podemos se maneja con los *WaiUs* que se tienen en la línea 4 y 5.

25. CONFIGURACIÓN DE PUERTO DE SALIDA CON LOS DISTINTOS SISTEMAS DE NUMERACIÓN

En el siguiente caso se repite el programa con la variante que los pines del puerto enciendan aleatoriamente. Se observa que, ahora, el puerto B se trabajó de manera binaria. Esta diferencia se consigue agregando el signo % y escribiendo el valor numérico deseado, equivalente a la configuración de esa misma naturaleza.

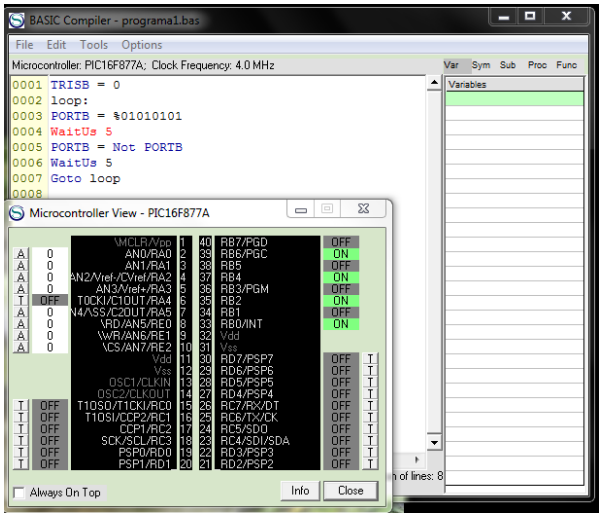
Las tablas de ayuda que se encuentra en el apéndice sirven para encontrar el valor en los distintos sistemas. En conclusión, indistintamente con qué sistema se trabaje este lo reconocerá.

Tabla IV. **Valores según su base numérica**

Decimal	Binario	Hexadecimal
85	1010101	55

Fuente: elaboración propia.

Figura 14. **Programa usando distintos sistemas de numeración**

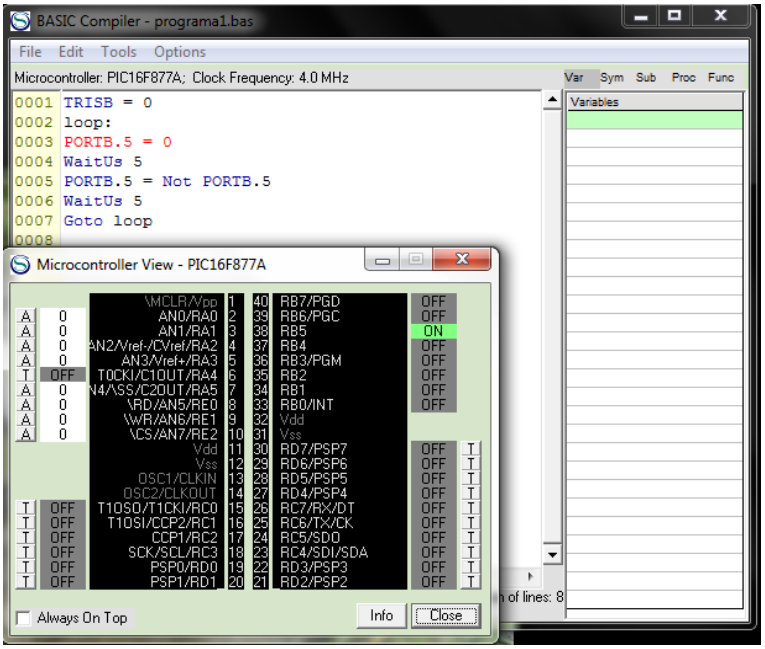


Fuente: elaboración propia.

26. CONFIGURACIÓN DE UN PIN COMO PUERTO DE SALIDA

Nuevamente se usa el ejemplo y se modifica para que únicamente un pin del puerto centelle.

Figura 15. **Configuración de un pin como puerto de salida**



Fuente: elaboración propia.

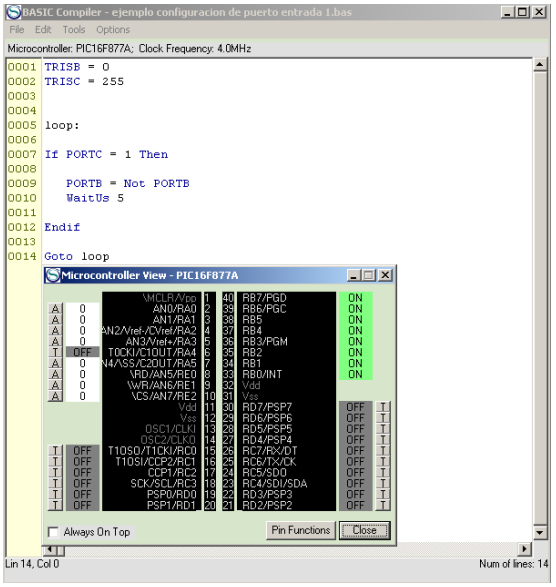
Como se puede observar, solo se modificó el comando PORTB, ahora tiene un punto seguido de un número, esto le indica al sistema que únicamente se está refiriendo a un pin específico del puerto.

27. CONFIGURACIÓN DE UN PIN COMO PUERTO DE ENTRADA

El principio fundamental del PIC es tener interacción con su entorno tomando decisiones a partir de datos que le son adquiridos, los PIC tiene diversidad de módulos de adquisición de datos, que son configurados por registros internos, no todos los puertos soportan dicho trabajo esta información se puede obtener del *DATA SHEET* de cada PIC.

Para el primer ejemplo se configura el puerto C como puerto de entrada, si el PIC recibe un pulso en el pin 0 de dicho puerto centellara el puerto B.

Figura 16. **Configuración de puerto de entrada**

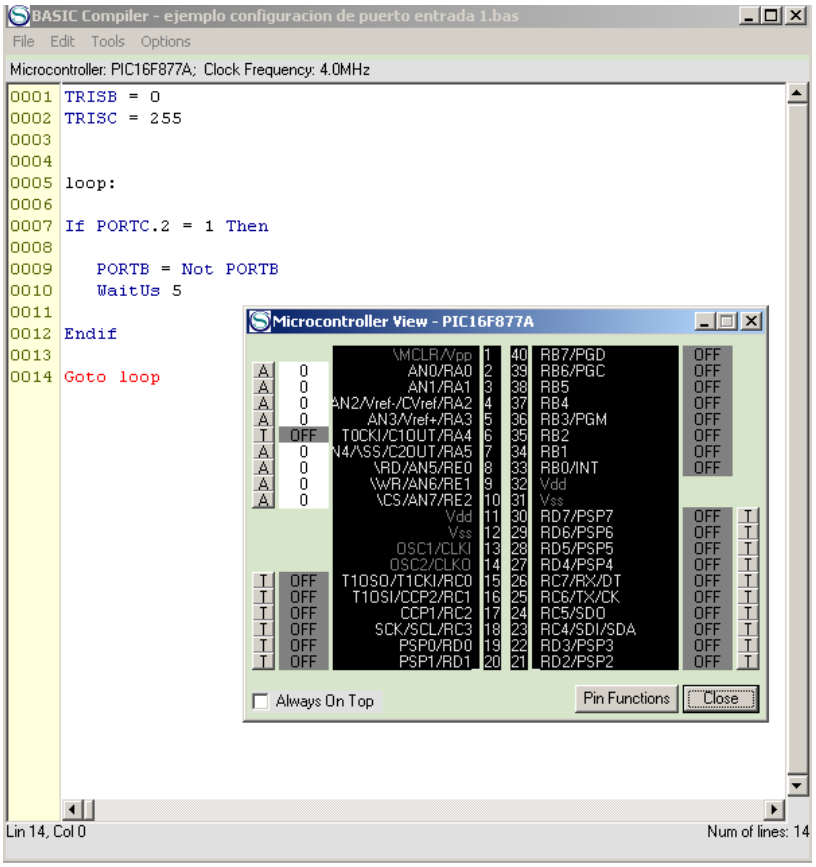


Fuente: elaboración propia.

28. CONFIGURACIÓN DE PUERTO DE ENTRADA / SALIDA

En el segundo ejemplo se configura para que el pulso venga del ping 2 del puerto C.

Figura 17. **Configuración de un pin como puerto de salida**



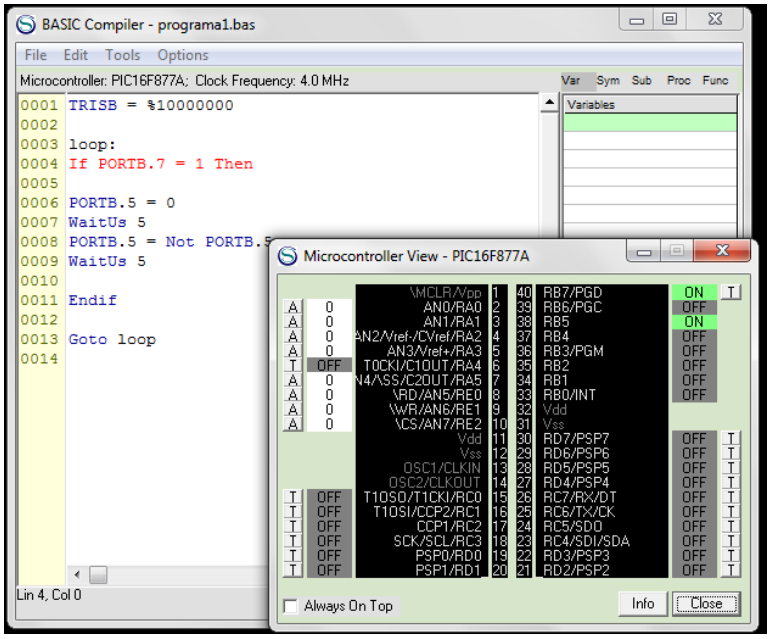
Fuente: elaboración propia.

29. CONFIGURACIÓN DE PUERTO DE ENTRADA OPTIMIZACIÓN DE PUERTOS COMO

En el siguiente ejemplo se configurar un puerto para que obtenga información y la saque de este.

Un puerto no obligatoriamente se debe configurar completamente como dispositivo de entrada o salida, pero se debe tener cuidado de declarar adecuadamente los pines.

Figura 18. **Optimización de puertos como dispositivos de e / s**



Fuente: elaboración propia.

Como se observa en la línea 1 el registro *TRISB* tiene un valor decimal 128 Hexadecimal (0X80), Binario (10000000). Este valor convierte el pin número 7 del puertoB en entrada digital y los demás pines como pines de salida digital.

Al tener un valor digital de 1 en el pin de entrada, el pin número 5 del mismo puerto empezará a centellar.

30. CONFIGURACIÓN DE PUERTO DE ENTRADA QUE POSEEN MÓDULOS ADC

En el siguiente ejemplo se explicará cómo configurar un puerto de entrada digital cuando tiene un MODULO Convertidor Análogo Digital (ADC). Cuando un puerto tiene un módulo convertidor se debe configurar un registro adicional al TRIS dicho registro es *ADCON1*. Este registro es el encargado de activar o desactivar el módulo convertidor en caso no se configure creará conflicto y el microprocesador no podrá trabajar adecuadamente.

Tabla V. **Configuración ADCON1 general**

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

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

A = Analog input D = Digital I/O
C/R = # of analog input channels/# of A/D voltage references

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf

Consulta: 8 octubre 2017.

Tabla VI. **Configuración Adcon1 Registro 9Fh**

ADCON1 REGISTER (ADDRESS 9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf

Consulta: 8 octubre 2017.

En la tabla se observa que existen N posibilidades de configurar un puerto de entrada. En la primera línea se pueden configurar todos los pines de un puerto como entrada analógica, también se pueden configurar como 3 pines digitales y 5 análogos, 7 digitales y 1 análogo, todos dos pines digitales y ninguno analógico, etc.

En los casos de pines analógicos se explicarán más puntualmente con los ejemplos del convertidor análogo digital (ADC). En el siguiente ejemplo explicaremos como se trabaja el registro ADCON1. En una configuración de puerto de entrada digital.

Tabla VII. **Configuración ADCON1**

ADCON1 REGISTER (ADDRESS 9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf

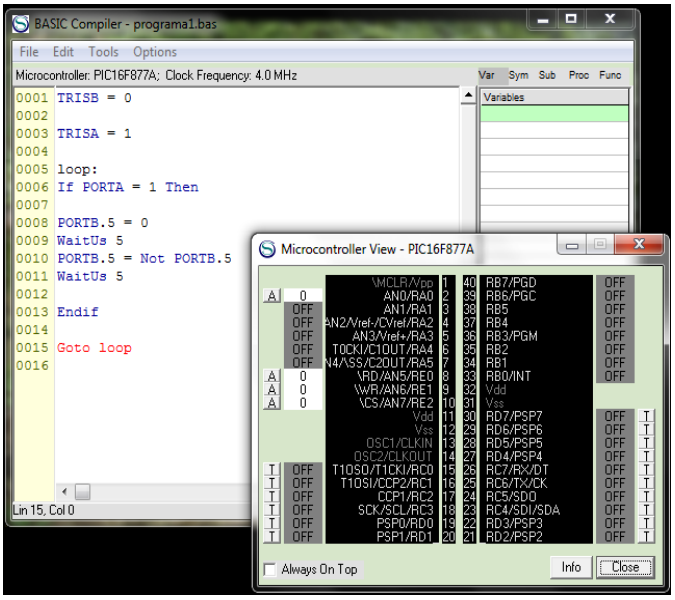
Consulta: 8 octubre 2017.

0 + 0 + 0 + 0 + 0 + 1 + 1 + 1 = 7

Las posiciones del registro ADCON 6 y 7 se coloca 0 ya que al colocar el puerto complemente digital quedan deshabilitadas las posiciones 4 y 5.Son posiciones que no se utilizan en este microprocesador, estas posiciones son utilizadas en los casos en donde ay más de 8 pines de entrada análoga.

Se configurará el puerto A ya que tiene modulo Convertidor Análogo Digital (ADC) como puerto de entrada. Para fines didácticos se muestra la imagen cuando no se configura el registro ADCON1.

Figura 19. **Configuración de puerto de entrada con módulos ADC**

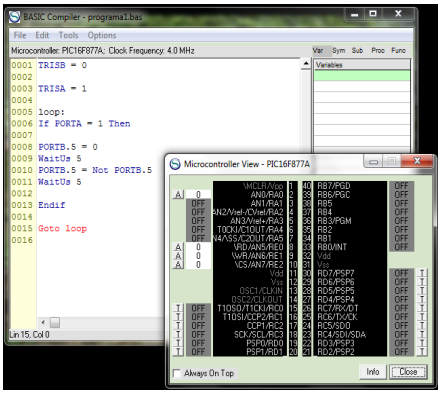


Fuente: elaboración propia.

Se puede observar en la imagen que el microprocesador el puerto A aparecen 7 pines con la etiqueta *OFF* y únicamente el pin de la posición 0 cuenta con un botón con la letra A que indica que es una entrada Analógica.

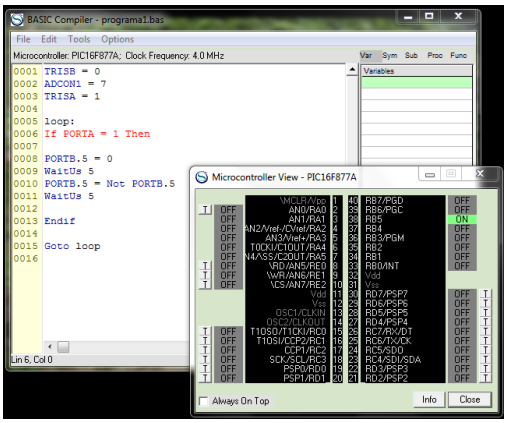
Cuando se configura correctamente el registro *ADCON1* el cambio es significativo.

Figura 20. **Configuración incorrecta de puerto con módulos ADC**



Fuente: elaboración propia.

Figura 21. **Configuración incorrecta de puerto con módulos ADC**



Fuente: elaboración propia.

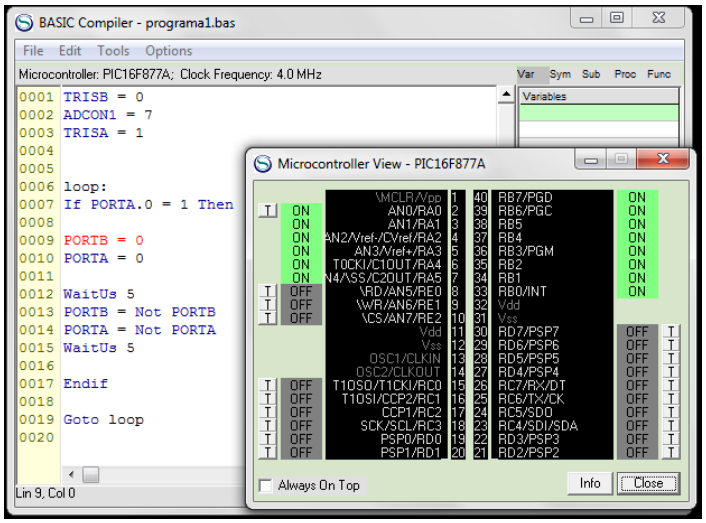
En la línea 2 del programa se observa la configuración del registro *ADCON1*, al comparar la imagen donde se configuro el registro y la que si está configurada se observa la importancia del registro *ADCON1*.

Indistintamente del registro *ADCON1,* se debe utilizar también el registro TRIS. El efecto de la configuración del registro *TRIS* será el mismo que en los casos anteriores.

Configurando el registro *ADCON1* como dispositivo digital y el registro *TRISA*=1 se configura el puerto A como el pin 0 como pin de entrada y del pin 1 al 7 como salida.

Figura 22. **Configuración de puerto de entrada con módulos ADC**

.

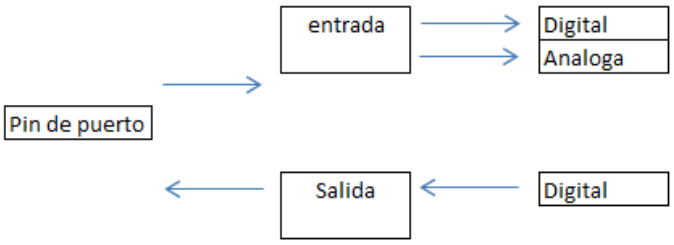


Fuente: elaboración propia.

31. CONFIGURACIÓN DE PUERTO DE SALIDA QUE POSEEN MÓDULOS ADC

De acuerdo con el ejemplo de configuración del registro *ADCON1*, en este se usará un PIC que tiene esas capacidades en el puerto A. Se deberá configurar el puerto, como puerto digital para luego indicarle si se desea como puerto de entrada o salida.

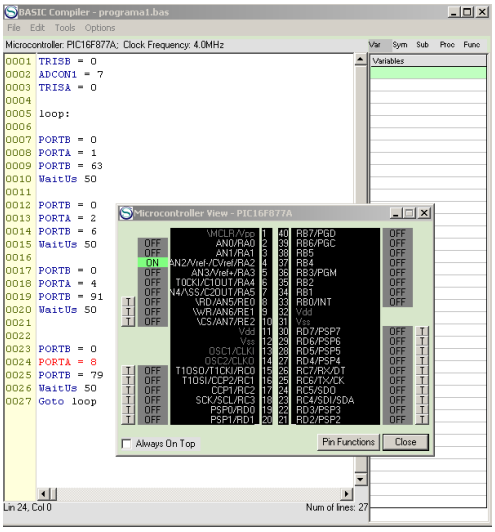
Figura 23. **Diagrama de funcionamiento puerto I/O D/A**



Fuente: elaboración propia.

Como se ve en el recuadro anterior un pin de puerto puede ser programado como entrada análoga o digital según la necesidad y como puerto de salida únicamente digital. Esto se debe dejar configurado para que el microprocesador sepa qué tipo de señal debe trabajar. Esta configuración la hace el registro *ADCON1* para el caso del microprocesador 16f87x. Para los demás microprocesadores se verificará la hoja de especificaciones.

Figura 24. **Programa puerto I/O D/A**



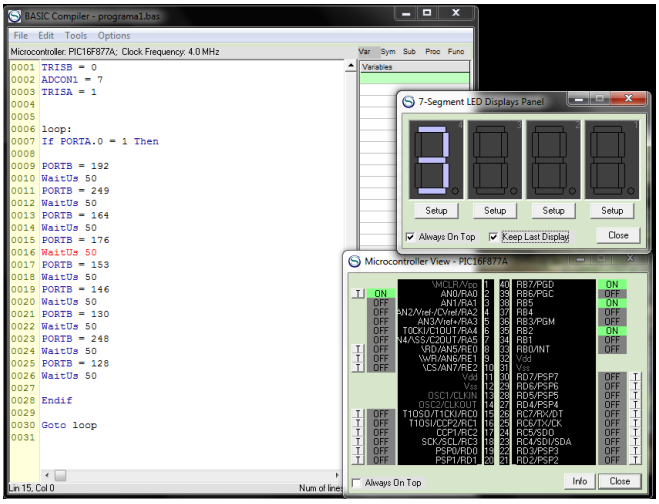
Fuente: elaboración propia.

32. CONFIGURACIÓN DE PUERTO DE SALIDA USO DE UN DISPLAY

En este apartado se explicarán varios casos donde se necesita interactuar con diversos dispositivos de salida.

Un display es un arreglo de led´s que ayudan a desplegar un valor numérico en un dispositivo que se entienda fácilmente. Para desplegar información en un display se debe relacionar cada pin del puerto que utilizaremos como dispositivo de salida con un segmento de dicho display para luego desplegar coherentemente el valor.

Figura 25. **Uso de un display**



Fuente: elaboración propia.

Línea 1 Configuramos el *TRIS* del puerto b como dispositivo de salida.

Línea 2 Configuramos el registro *ADCON1* para que el puerto A sea puerto digital

Línea 3 Configuramos el *TRIS* del puerto A, pin 0 como dispositivo de entrada.

Línea 4 Etiqueta que servirá para realizar el ciclo infinito del programa.

Línea 5 Línea que sirve para comparar el valor del pin 0 del puerto A de ser válida la comparación entrará al realizar las líneas 9 al 26. Si es falsa se saltará hacia la línea 28. Entre cada despliegue de datos se da un tiempo de retardo para observar el valor en el display. Se debe recordar que el microprocesador trabaja a velocidades muy rápidas que el ojo humano no es capaz de distinguir un cambio de valor.

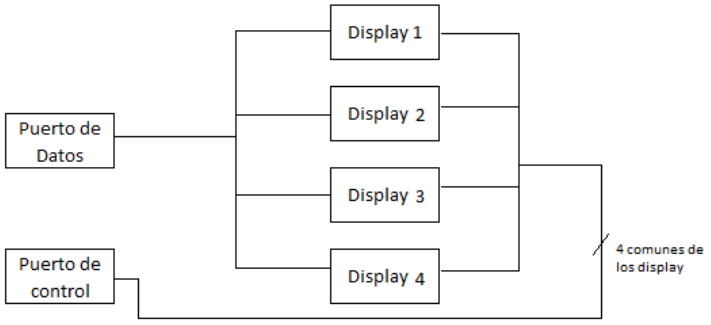
Línea 6 Fin del ciclo de comparación

Línea 7 Fin del programa y punto de enciclaje del mismo.

33. CONFIGURACIÓN DE PUERTO DE SALIDA USO DE MATRIZ DE DISPLAY

En el siguiente ejemplo, el inconveniente será extraer la información en 4 display independientes. El microprocesador 16f877a tiene 4 puertos con 8 pines y un puerto con 4 pines. El inconveniente de usar los 4 puertos de 8 pines es que si se desea utilizar el *pic* para algo más que desplegar los datos al display, los puertos serán limitados por ello se utilizará el método de multiplexación que es la habilitación por etapas de la información hacia un mismo destino.

Figura 26. **Diagrama de matriz de display**



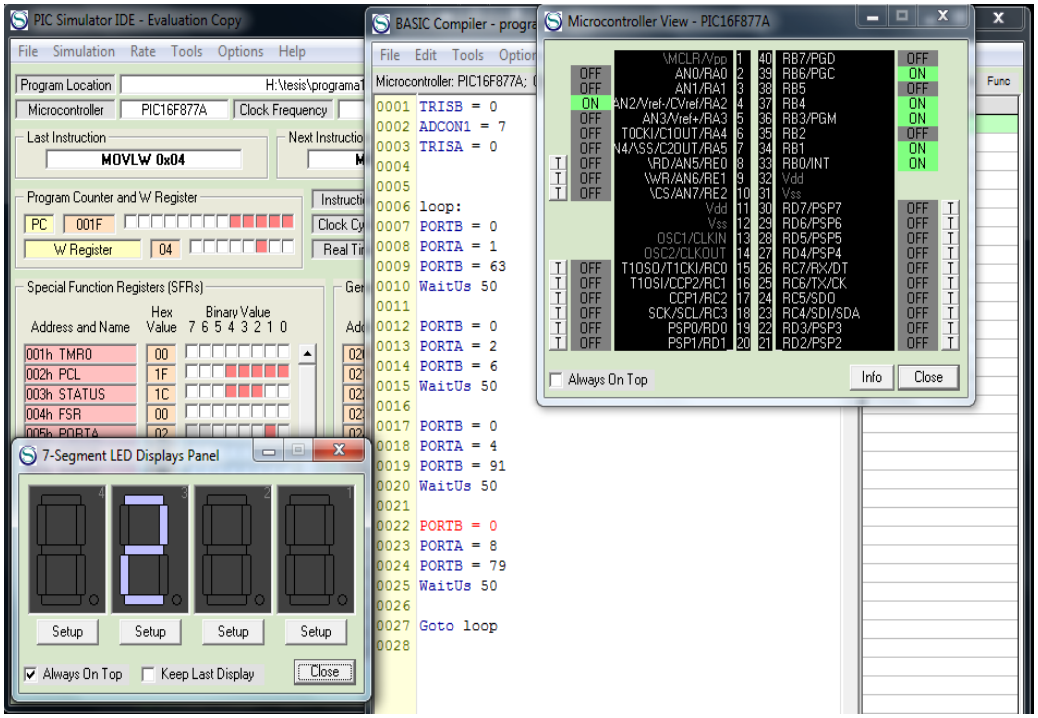
Fuente: elaboración propia.

Se programará un puerto como puerto de control el cual llevará los datos que se desplegarán. Será un puerto común para los 4 display y un puerto de control el cual encenderá el display indicado para mostrar la información.

El puerto de control siempre tendrá únicamente encendido un display el cual mostrará el valor. Si en el puerto de datos hay datos del display 1, en el

puerto de control habrá una habilitación al común del display 1. Esto provocará que se cierre el circuito del display 1 y se muestren los datos mientras los circuitos de los otros displays están abiertos provocando que no se vea información en ellos.

Figura 27. **Programa matriz de display**



Fuente: elaboración propia.

34. CONFIGURACIÓN DE PUERTO DE SALIDA USO DE MOTOR PASO A PASO

Un motor de paso es un dispositivo mecánico que convierte un pulso eléctrico en movimiento controlado. Hay dos maneras de programar un motor de paso torque o velocidad. Un microprocesador no es capaz de hacer trabajar un motor de paso por si solo necesita de una etapa de potencia, calculada para los requerimientos del motor que trabajemos.

Dependiendo de las fases y del tipo del motor así serán los pines que se deben configurar como puertos de salida digital.

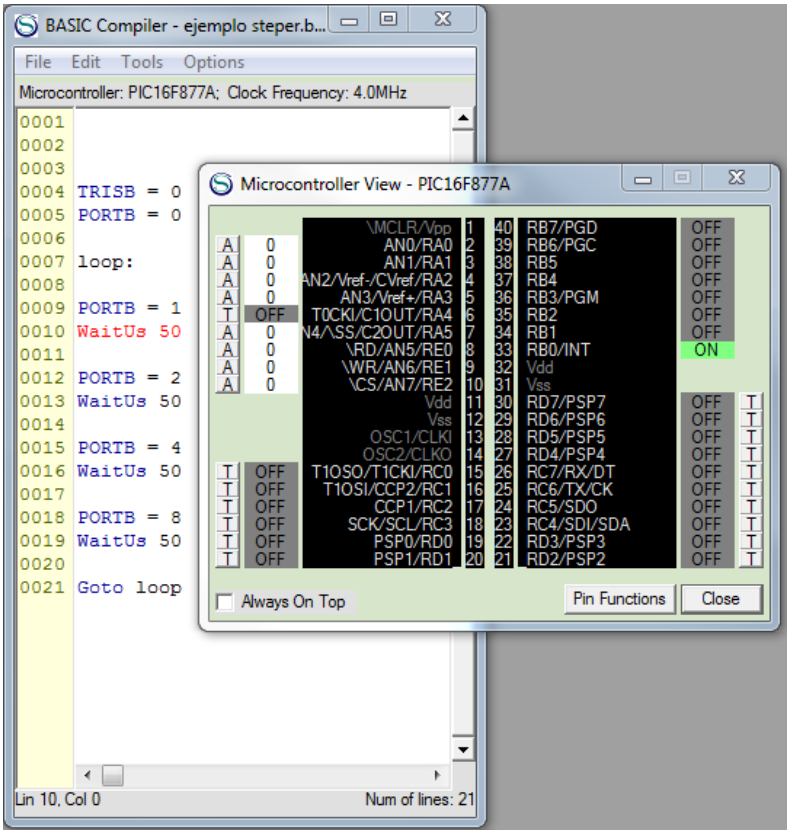
Básicamente hay 2 tipos de motores de paso:

- Motores de paso a paso unipolares
- Motores de paso a paso bipolares

Los motores unipolares suelen tener 5 o 6 cables de salida dependiendo de su conexión interna. Este tipo más simple de controlar, utilizan un cable común a la fuente de alimentación y, posteriormente se colocan las otras líneas a tierra en un orden específico para generar cada paso. Si tienen 6 cables es porque cada par de bobinas tienen un común separado, si tiene 5 cables es porque las cuatro bobinas tienen un polo común; un motor unipolar de 6 cables puede ser usado como un motor bipolar si se deja las líneas del común al aire.

Los motores bipolares tienen, generalmente, 4 cables de salida. Necesitan ciertos trucos para ser controlados debido a que requieren del cambio de dirección de flujo de corriente a través de las bobinas en la secuencia apropiada para realizar un movimiento.

Figura 28. **Uso de motor paso a paso**



Fuente: elaboración propia.

El programa no es muy complicado ya que únicamente se debe configurar la secuencia de salida y tomar un tiempo para pasar al siguiente pulso. En este punto hay un inconveniente ya que un microprocesador trabaja a frecuencias de trabajo superiores a los motores de paso. Por este motivo, se debe calcular el

tiempo que tarda en energizar la bobina del motor de paso para dar el siguiente impulso.

El PIC manda los pulsos al conjunto de transistores que forman parte del área de potencia. Estos suministran la corriente para las bobinas del motor.

35. CONFIGURACIÓN DE PUERTO DE SALIDA USO DE MOTOR DC CON DOBLE GIRO

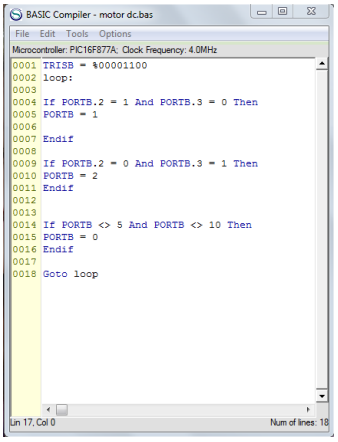
Parte de los requerimientos para usar un motor *DC* es que pueda cambiar de giro. Esto se logra con un circuito H como circuito de potencia y de control.

En anexos se incluye un circuito construido con transistores, también hay una solución que consta de un integrado que facilita la construcción.

En la tabla de funcionamiento se evidencia que cuando se envían dos pulsos del mismo valor digital, el motor no gira, por lo tanto, siempre se debe tener como salida de puerto 0 a menos que se desee que gire el motor.

Se usarán los pines 0 ,1 del puerto b para esta actividad como puertos de salida y los pines 2 y 3 como puertos de entrada, con esto se practicará la configuración de un solo puerto como entrada y salida.

Figura 29. **Uso de motor con doble Giro**



Fuente: elaboración propia.

En la línea 1 se configura el puerto según los requerimientos establecidos pines 0 y 1 como puertos de salida y los pines 2 y 3 como entrada.

Se crea una etiqueta que nos servirá para crear un *loop* en el programa.

Comparamos los estados de los pines 2,3 y, según su estado se les da el valor de 1 o 2.

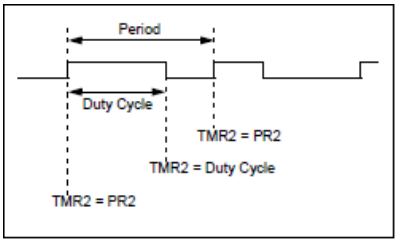
La comparación de la línea 14 se encarga de que haya giro en el motor una vez solo haya un pin de entrada activo.

La pregunta es por qué esos dos valores 5 y 10. Cuando se consulta sobre el valor de un puerto entero, la consulta se hace sobre el registro entero del puerto el cual contiene tanto los valores externos como internos.

36. CONFIGURACIÓN DE PUERTO DE SALIDA CONTROL DE VELOCIDAD MOTOR *DC* (PWM)

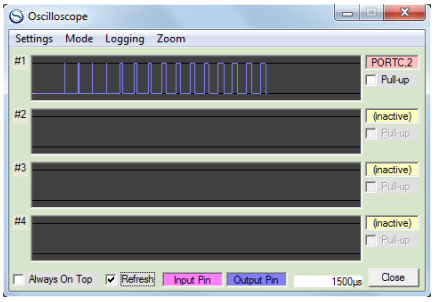
En este punto se utiliza el primer módulo interno del *PIC*. Dicho modulo es el PWM. Es modulación por ancho de pulso y controlándolo se controla la velocidad de un motor en *DC*, ya que las bobinas del motor estarán excitadas por un tiempo determinado constantemente que también puede ser calculado con el voltaje eficaz inyectado al motor *DC*.

Figura 30. **Ciclo de pulso PWM**



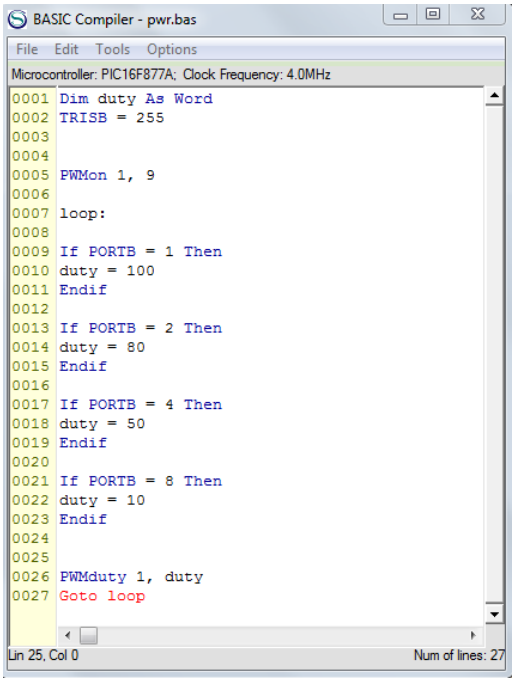
Fuente: elaboración propia.

Figura 31. **Variación del ciclo de pulso PWM**



Fuente: elaboración propia.

Figura 32. **Variación del ciclo de pulso PWM**



Fuente: elaboración propia.

En la primera línea del programa es para declarar una variable *Duty* con la cual se variará la duración del tiempo en alto del tren de pulsos.

Los módulos PWM internos (más precisamente: los modos PWM de los módulos CCP) se activan con la instrucción PWMON. El primer argumento es número de módulo y debe ser una constante en el rango 1-3. El segundo argumento se utiliza para la selección de modo.

Tabla VIII. **Configuración de pulso PWM**

mode 1	10-bit	244Hz
mode 2	10-bit	977Hz
mode 3	10-bit	3906Hz
mode 4	9-bit	488Hz
mode 5	9-bit	1953Hz
mode 6	9-bit	7813Hz
mode 7	8-bit	977Hz
mode 8	8-bit	3906Hz
mode 9	8-bit	15625Hz
mode 10	7-bit	1953Hz
mode 11	7-bit	7813Hz
mode 12	7-bit	31250Hz

Fuente: elaboración propia.

37. CONFIGURACIÓN DE PUERTO DE SALIDA COMUNICACIÓN SERIAL (USART)

El acrónimo USART hace referencia al receptor transmisor síncrono asíncrono universal. Es una comunicación serial síncrona de 8 o 9 bit según la configuración establecida previamente. En el caso el PIC16F877A tiene el módulo USART en el puerto C pines 6 y 7, los cuales deben configurarse antes de usarlos.

Su principal desventaja es la conexión limitada a un servicio de transmisión punto a punto. La distancia máxima es de 50 pies, es una comunicación muy sencilla donde únicamente se necesitan tres elementos de conexión entre el equipo: transmisor (TX), receptor (RX) y la tierra o común entre circuitos. Existen dos tipos de comunicación serial: síncrona y asíncrona.

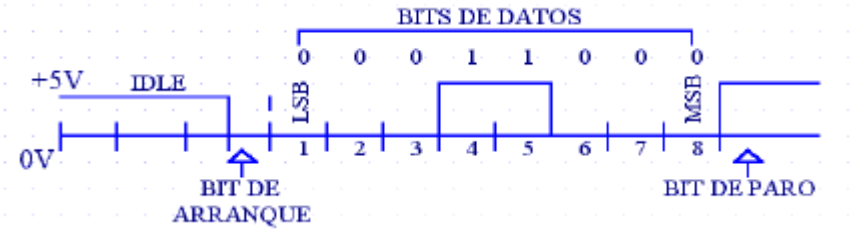
En la comunicación asíncrona no es necesario enviar el pulso de reloj ya que la duración de cada bit está determinada por la velocidad con que se realiza la transferencia de datos.

Una trama de comunicación asíncrona está compuesta por varios elementos

- Bit de arranque: Es el bit que indica el inicio de la información.
- Bit de paro: Es el bit que indica el final de la información
- Bit LSB: Bit menos significativo de la información enviada
- Bit MSB: Bit más significativo de la información enviada

Estado *IDLE*: Este estado indica que el equipo está listo para enviar información.

Figura 33. **Comunicación serial (USART) tren de pulso**



Fuente: elaboración propia.

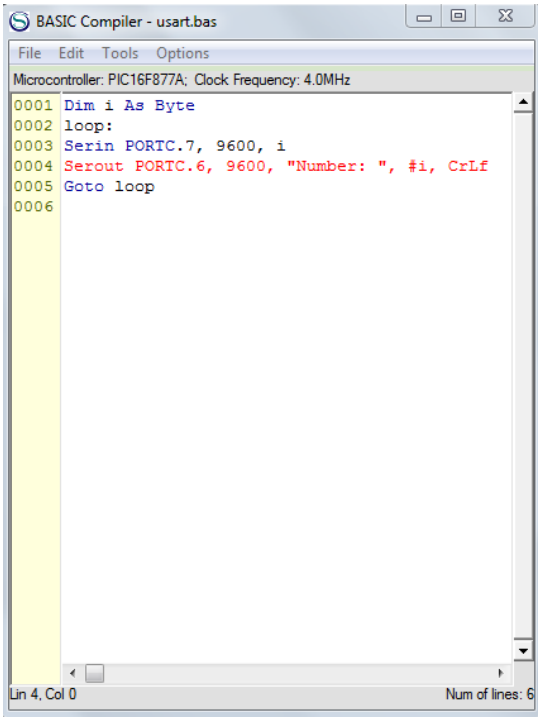
Es importante que los elementos de transmisión (Tx) y recepción (Rx) tengan los mismos parámetros de trabajo, de lo contrario, los paquetes nunca serán entregados/enviados de una manera legible para el sistema.

Para distancias cortas los niveles lógicos de voltaje (0-5V) son viables, pero a mayor distancia entre los elementos *Tx* y *Rx* aumenta la distorsión debido a los efectos capacitivos y resistivos del medio de comunicación, donde el efecto resistivo aumenta a razón de la distancia y el efecto capacitivo aumenta en relación con la velocidad de transmisión. Hay que tener en cuenta para los valores lógicos de 0 el valor de voltaje debe ser menor a 0,8V y para valores lógicos de 1 el voltaje debe ser mayor a 2V. Para solucionar este inconveniente se integra al sistema un circuito que eleva el voltaje a valores de +/- 12V sin necesidad de fuente bipolar, El circuito Max232 que es un circuito que cumple con las normas RS-232. Las normas RS-232 son:

Un “1” lógico es un voltaje comprendido entre –5v y –15v en el transmisor y entre -3v y –25v en el receptor.

Un “0” lógico es un voltaje comprendido entre +5v y +15 v en el trasmisor y entre +3v y +25 v en el receptor.

Figura 34. **comunicación serial (*USART*)**



Fuente: elaboración propia.

Este ejemplo es similar a un espejo, se recibe lo que se envia.

La línea 1 se declara la variable que se utilizará como contenedora de la información que recibe. Se crea una etiqueta que nos servirá como *loop.*

El comando *Serin* sirve para seleccionar el puerto y pin que se utiliza como TX, el 9600 es la velocidad de transmisión. El comando Serout sirve para seleccionar el puerto y pin que se utiliza como RX, el 9600 es la velocidad de transmisión.

Se debe prestar atención a la velocidad a la que se está trabajando ya que si se trabajan distintas velocidades no será posible la sincronización de trama de datos y, por consiguiente, no será posible la comunicación entre los dos periféricos.

38. CONFIGURACIÓN DE PUERTO DE SALIDA CONFIGURACIÓN DE LCD

Con el ejemplo de desplegar la información de salida por medio de un display nos topamos con el inconveniente que no todas las letras pueden ser escritas de una manera legible, para darle solución a este inconveniente vamos a escribir un programa que su salida de información sea atreves de una pantalla LCD, el acrónimo LCD viene de Pantalla de Cristal Líquido. Dependiendo de la configuración de la pantalla LCD se pueden visualizar 2 líneas de 16 caracteres cada una, es decir, 2x16=32 caracteres.

La LCD dispone de una matriz de 5x8 puntos para representar cada carácter. En total se pueden representar 256 caracteres diferentes. 240 caracteres están grabados dentro del LCD y representan las letras mayúsculas, minúsculas, signos de puntuación, números, etc... Existen 8 caracteres que pueden ser definidos por el usuario. Esto es una gran ventaja si se desea personalizar un carácter:

Las características principales de las LCD son:

- Consumo muy reducido, del orden de 7.5mW .
- Pantalla de caracteres ASCII, además de los caracteres japoneses.
- Kanji, caracteres griegos y símbolos matemáticos.
- Desplazamiento de los caracteres hacia la izquierda o a la derecha.
- Memoria de 40 caracteres por cada línea en pantalla, visualizándose 16 caracteres por línea.

- Movimiento del cursor y cambio de su aspecto · Permite que el usuario pueda programar 8 caracteres.
- Pueden ser gobernados de 2 formas principales: o conexión con bus de 4 bits o conexión con bus de 8 bits.

Las LCD poseen 3 módulos de memoria: El primero es el *DDRAM* (*Data Display RAM*) donde se almacenan los caracteres que se muestran en la pantalla.

El segundo módulo es llamado *CGROM*, es un espacio de memoria no volátil donde se almacena 192 caracteres que pueden ser visualizados, cada carácter tiene una representación binaria de 8 bits.

En el tercer módulo de memoria *CGRAM* (*Character Generator RAM*) se almacenan los 8 caracteres definidos por el usuario. Esto permite personalizar los mensajes. La tabla de caracteres se puede ver en los anexos.

Las pantallas LCD constan de varios pines que se necesitan configurar en el PIC para que se puede comunicar con la LCD.

Básicamente los registros se pueden separar como registros de datos y control.

Las pantallas LCD pueden ser de 4 u 8 bits de datos. Si se tuviera una pantalla de 4 bit se optimizarían los puertos del PIC para que únicamente un puerto controle la pantalla LCD. De lo contrario, se debe dedicar un puerto únicamente para datos hacia la LCD. Esto se complica cuando se usa PIC de únicamente 4 puertos. Para este ejemplo, se utiliza una LCD con un bus de datos 4 bit.

Si configuramos la pantalla para trabajar con un bus de datos de 4bit debemos tomar los 4 más significativos del bus para trabajar y los 4bits restantes deben ser llevados a un 0 lógico.

Cuando se energiza la LCD debemos se debe tomar un tiempo mayor a los 10 ms antes de enviar algún dato a la LCD. Esto se debe a que es el tiempo aproximado que tarda el microprocesador de la LCD en auto inicializarse después de ser energizado.

Comandos básicos para trabajar una LCD son:

- *Clear Display*

Borra la información almacenada en la *DRAM* y coloca el cursor en la primera posición (dirección 00H) y queda a la espera de información en el bus de datos.

Tabla IX. **Configuración de registros LCD**

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

Fuente: elaboración propia.

- *Entry Mode*

Set: Establece la dirección de movimiento del cursor y especifica si la visualización se va desplazando a la siguiente posición de la pantalla o no.

Tabla X. **Configuración de registros LCD**

S	/W	B7	B6	B5	B4	B3	B2	B1	B0

Fuente: elaboración propia.

I/D=1 Incrementa la dirección del cursor

I/D=0 Decrementa la dirección del cursor

S=1 Desplaza la visualización cada vez que se escribe un dato

- Home

En la *DDRAM* coloca el cursor en la dirección 00H, sin borrar el contenido de registro, regresa el cursor al carácter 1 de la *LCD*.

Tabla XI. **Configuración de registros LCD**

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	0

Fuente: elaboración propia.

- *Display*

ON/OFF Control: activa o desactiva la pantalla LCD (D) y/o el cursor (C), establece si el cursor parpadea o no (B).

Tabla XII. **Configuración de registros LCD**

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	D	C	B

Fuente: elaboración propia.

- *Function Set*

Establece el tamaño del bus de datos con que se trabajará, número de líneas y tipo de carácter

Tabla XIII. **Configuración de registros LCD**

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	x	x

Fuente: elaboración propia.

DL=1 Trabaja con el bus de datos de 8bits

DL=0 Trabaja con el bus de datos de 4bits

N=1 la presentación de datos se hace en la línea 1

N=0 la presentación de datos se hace en la línea 2

F=1 Caracteres de tamaño 5x10 puntos

F=0 Caracteres de tamaño 5x7 puntos

- DDRAM ADDRES SET

Establece la dirección de la memoria de datos DDRAM a partir de la cual se almacenan los datos por visualizar.

Tabla XIV. **Configuración de registros LCD**

S	/W	B7	B6	B5	B4	B3	B2	B1	B0
			Dirección de memoria de datos DDRAM						

Fuente: elaboración propia.

- Ready Busy Flag and Address

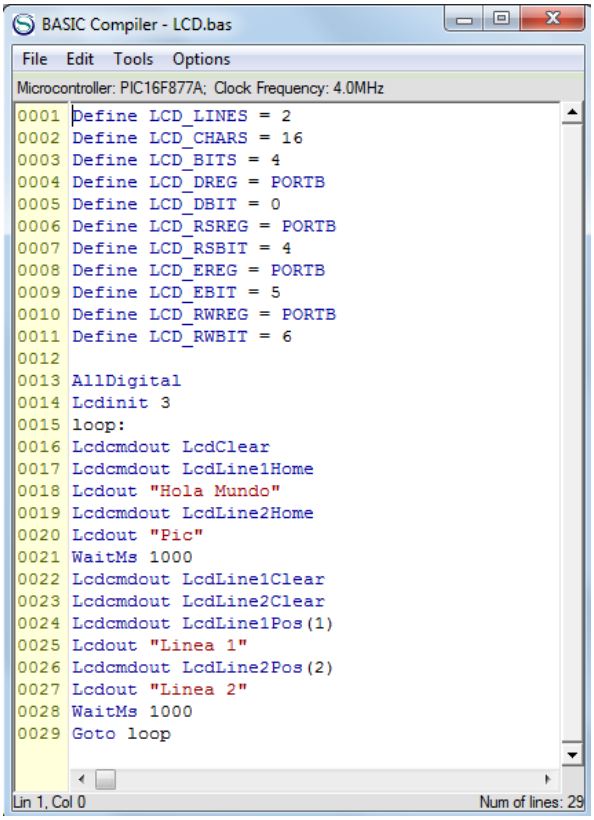
Lectura de bandera *busy*(BF) e indica la última dirección empleada de la DDRAM o CGRAM.

Tabla XV. **Configuración de registros LCD**

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BF	Dirección de memoria de datos DDRAM						

Fuente: elaboración propia.

Figura 35. **Programa de configuración de LCD**



Fuente: elaboración propia.

Línea 1 Se define que la cantidad de líneas que posee la pantalla.

Línea 2 Se define la cantidad de caracteres que posee la pantalla.

Línea 3 Se define la cantidad de bit de datos con que trabaja la pantalla.

Línea 4 Se define el puerto que estará a cargo del registro de datos. En este caso es el puerto B.

Línea 5 Se configura a partir de qué posición del puerto empieza el bus de datos.

Línea 6 Se define el puerto que estará a cargo del registro de Rs en este caso es el puerto B. y se le indica qué posición del puerto

trabajará dicho registro. Así se continúa con los demás registros de control.

Línea 7 Se inicializa la pantalla.

Línea 8 Se crea una etiqueta que servirá para el ciclo de programación.

Línea 9 Con el comando *Lcdcmdout* se le envía la orden de qué hacer a la pantalla LCD.

Línea 10 Con el comando *Lcdout* se le manda a la pantalla que escriba, no sin antes indicar en qué posición debe desplegar la información.

Línea 11 Se apunta a una nueva dirección del registro de la LCD.

39. CONFIGURACIÓN DE PUERTO DE ENTRADA / SALIDA VARIACIÓN DE PULSO PWM USANDO UN ADC

Un conversor o convertidor de señal analógica a digital (Conversor Analógico Digital, CAD; *Analog-to-DigitalConverter*, ADC) es un dispositivo electrónico capaz de convertir una señal analógica, ya sea de tensión o corriente, en una señal digital mediante un cuantificador y codificándose en muchos casos en un código binario en particular. Donde un código es la representación unívoca de los elementos. En este caso, cada valor numérico binario hace corresponder a un solo valor de tensión o corriente.

En la cuantificación de la señal se produce pérdida de la información que no puede ser recuperada en el proceso inverso, es decir, en la conversión de señal digital a analógica. Esto se debe a que se truncan los valores entre 2 niveles de cuantificación, a mayor cantidad de bits, mayor resolución y por lo tanto menor información perdida.

Un ADC convierte el voltaje V que tiene en un pin (que tendrá que estar declarado como entrada con el correspondiente registro TRISA)) y lo convierte en un número. El voltaje se mide de acuerdo con un voltaje mínimo, Vref(-) , y con un voltaje máximo, Vref (+).

$$V\_norm = (V - Vref(-)) / (Vref(+) - Vref(-))$$

La fórmula anterior corresponde a un voltaje normalizado. Si el voltaje V alcanza el máximo (Vref+) se tendrá una salida de 1 y si se queda en el mínimo (Vref-) una salida de 0.

Normalmente, Vref- suele ser Vss=GND=0V y Vref+ = Vcc = 5V, pero pueden usarse otros voltajes de referencia. Si, por ejemplo, se desea medir una señal cuya oscilación se sabe que está entre 2 y 3 voltios se usaría Vref-=2 y Vref+=3. Así se aprovecharía mejor el rango dinámico del conversor.

Como el microcontrolador no manejará números en coma flotante, el voltaje normalizado se expresa con un entero, convirtiendo el intervalo real [0,1] en el intervalo de niveles enteros entre [0 y Nmax-1]. La resolución del *ADC* es una característica fundamental y expresa el número de niveles con los que se cubre el intervalo [0,1]. Por ejemplo, en los *PIC,* generalmente, se tiene una resolución de 10 bits, que representan 2^10=1024 niveles. El intervalo real [0,1) se aplicaría al intervalo [0,1023]. Si se asume un rango de 5V, se tendrá que la resolución de cada nivel es de r=5/1024 V=4.88 mV. Según la documentación de microchip (esto puede variar para otros microcontroladores) cualquier voltaje entre [0 y r] (o por debajo de 0, lo que corresponde a V<*Vref*-) se cuantificaría en el nivel 0. Entre r y 2r tendríamos una salida de nivel 1. Así hasta llegar a nivel 1023 que cuantificaría voltajes por encima de 1023r = 1023x 5/1024 = 4.995V. Como se ve, voltajes por debajo de *Vref*- o por encima de *Vref*+ son posibles y se cuantifican como nivel mínimo 0 o máximo, 1023. Niveles por debajo de 0V o por encima de la tensión de alimentación (normalmente 5V) pueden dañar el PIC.

Tabla XVI. **Cuantificar y codificar**

Nivel	Voltaje
0	<r
1	[r,2r]
2	[2r,3r]
…	…
1023	>1023r

Fuente: elaboración propia.

Aunque un PIC puede tener del orden de 8-12 posibles canales (pines) de entrada analógica, solo tiene normalmente un único módulo ADC, lo que significa que no se pueden tomar medidas simultáneas de varios canales. Si es necesario, se pueden conectar (seleccionar) los sucesivos canales al ADC para medir sus voltajes.

El proceso de una conversión ADC se divide en un tiempo de adquisición Ta (durante el cual un condensador interno se carga al voltaje exterior) y un tiempo de conversión Tc (durante el cual se desconecta el pin exterior y se cuantifica el voltaje del condensador).

El tiempo de adquisición Ta depende de las características eléctricas del PIC (en particular de la capacidad del condensador). Si no se respeta este tiempo, el condensador no habrá alcanzado el nivel del voltaje exterior y la medida será incorrecta. Los *datasheet* de los PIC indican los Ta recomendados para diversas familias.

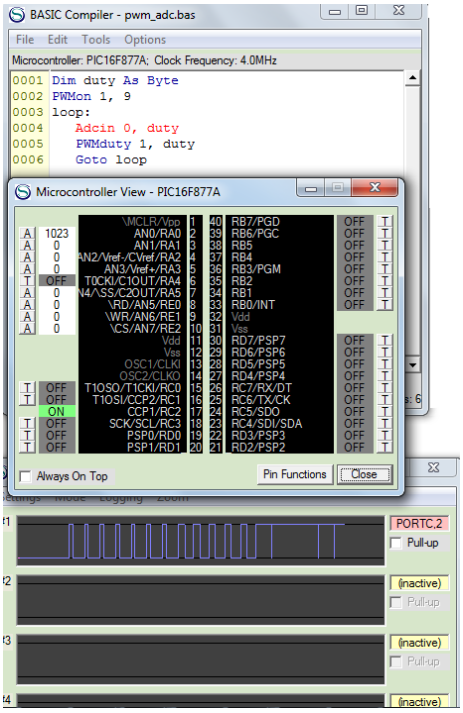
El tiempo de conversión Tc depende fundamentalmente del número de bits del conversor. La unidad básica es el llamado *Tad*, aproximadamente el tiempo necesario para ganar un bit adicional. La conversión total tarda entre 11 y 14 *Tad* (contando con la descarga final del condensador para estar listo para otra medida. El reloj del ADC se debe ajustar (como una fracción del oscilador principal) para que dicho *Tad* no sea inferior a un valor mínimo especificado en los *datasheet*. Por ejemplo, para la familia PIC18F252/452 el *Tad* mínimo es de 1.6 *usec* y Tc = 14 Tad. En cambio, para la familia PIC18F2520/4520 se tiene un *Tad* mínimo de 0.75 *usec* y un Tc = 11 *Tad*.

Cuando se trabaja con el ADC se debe configurar dos registros(ADCON0, ADCON1) y la información se obtiene en dos registros (ADRESH, ADRESL) dependiendo de la cantidad de bit de resolución que tiene nuestro ADC.

Un ejemplo sencillo y la solución para los problemas de los tiempos de trabajo de los motores paso a paso y el controlador de velocidad de un motor DC. Con la ayuda del *ADC* se puede variar el valor de este tiempo sin la necesidad de tocar la programación nuevamente. Si se cambia el modelo de motor se debe calcular de nuevo el tiempo que se tarda en energizar las bobinas del motor. Esto implica tocar la programación, pero con la ayuda de un potenciómetro se coloca en la entrada analógica se puede convertir un valor de voltaje en un tiempo variable, de esta manera se ajusta la velocidad en los motores.

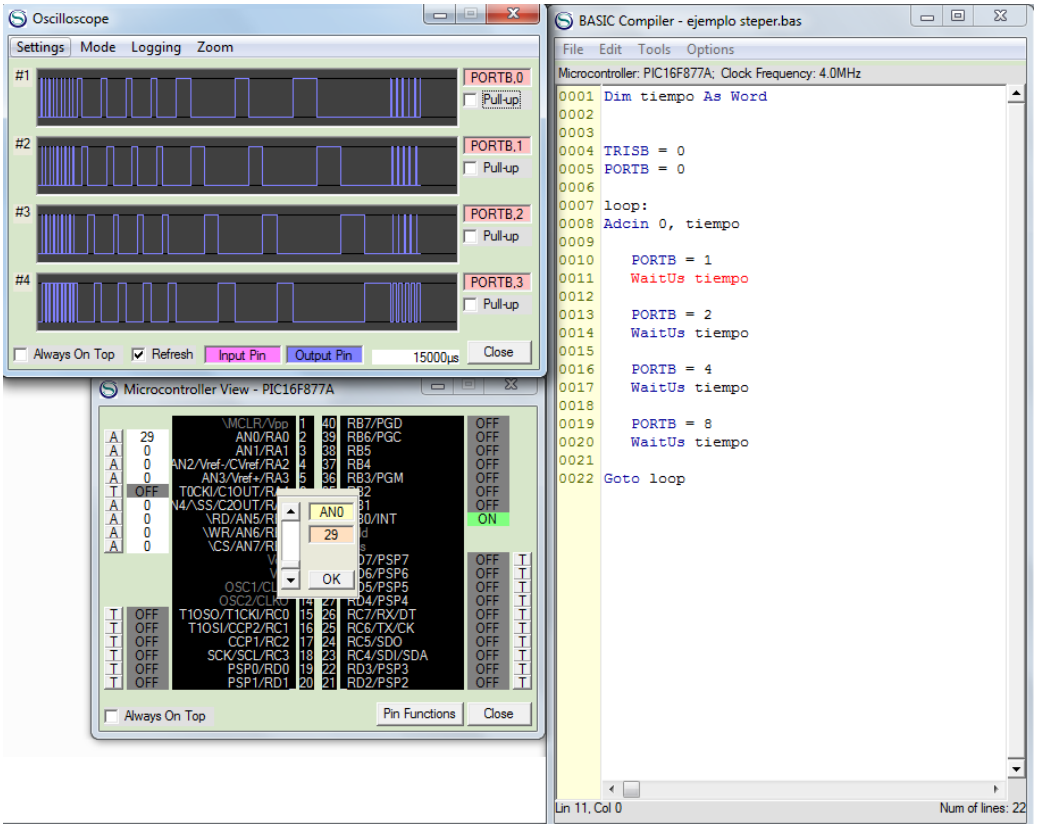
Únicamente hay que agregar a los ejemplos anteriores un comando, el cual activa el puerto ADC sin necesidad de configurar *ADCON0*, *ADCON1.* Solo se especifica el puerto que se desea usar como entrada digital y la variable donde se manejará el valor obtenido por el ADC.

Figura 36. **Pulso PWM variable con el ADC**



Fuente: elaboración propia.

Figura 37. **Pulso PWM variable con el ADC**



Fuente: elaboración propia.

Es claro que en ambos ejemplos el valor del pulso es significativamente diferente y esto repercute en la velocidad con que trabajar el motor.

Como parte de las prestaciones de un PIC algunos poseen el módulo de comparador analógico, con este módulo se comparan las señales analógicas y a partir de ellas se toman decisiones de procesamiento. En el caso del PIC 16F877A los pines de entrada para la señal que se comparará están multiplexados en el puerto A Pines del 0 al 3 y las salidas están en el mismo puerto, pero son los pines 4 y 5.

Cuando tenemos un PIC con módulo de comparación de voltaje integrado se puede usar un voltaje de referencia interno o también se puede tomar un voltaje de referencia externo, aunque esto representa que no se puedan utilizar pines.

Otra posibilidad es usar niveles de tensión configurables generados por el módulo de referencia de tensión.

Primero se debe configurar el registro CMCON. Este trabaja el funcionamiento de los comparadores. Hay 8 maneras de configurar las entradas y salidas de los comparadores de voltaje y se seleccionan con los 3 bit menos significativos del registro, también en este registro se puede leer el valor lógico del resulta de las comparaciones realizadas y finalmente nos permite invertir o no el resultado lógico de la comparación.

Hay que tener en cuenta que el registro de configuración del puerto A (*TRISA*) controlará la dirección de los datos sin importar la configuración que se seleccionen de los comparadores. Los pines pueden estar configurados como puertos digitales, pero no estarán en uso. Por consiguiente, esto repercute en un gasto de energía ya que consumirá más corriente que la especificada para una entrada analógica.

Si la entrada analógica VIN+ es mayor que la entrada analógica VIN-, entonces la salida del comparador será un estado digital alto.

Si la entrada analógica VIN+ es menor que la entrada analógica VIN-, entonces la salida del comparador será un estado digital bajo.

Las referencias de trabajo pueden ser internas o externas. Si se decide usar una referencia externa, esta no puede ser mayor al voltaje de alimentación

o común, según sea el caso. Si en cambio, se usa una referencia interna se debe configurar esta referencia en un módulo aparte compuesto por un conjunto de resistencias conectadas en serie que generan un divisor de voltaje.

La configuración del módulo referencia de voltaje de comparación se hace a través del registro *CVRCON*. Con este registro se configura, si el voltaje de referencia esta encendido o no y se ahorra energía.

La escalera de resistencia está separada para proporcionar dos rangos de valores *CVREF*, La referencia del comparador tensión de alimentación (CVRSRC) directamente de VDD. Cabe señalar que la tensión en la parte superior de la escalera es *CVRSRC* - *VSAT*, donde *VSAT* es la tensión de saturación de la potencia interruptor transistor. Esta referencia será solamente los valores de *CVRSRC* y *VSAT*. La salida del generador de referencia puede estar conectada al pin *RA2* / *AN2* / *VREF-* / *CVREF*. Esto puede ser utilizado como una simple función de convertidor de voltaje, en el caso que haya impedancia muy alta de carga.

40. PROGRAMAS REALIZADOS EN LENGUAJE ASM

En este módulo se programará un PIC en lenguaje *Asembler* (ASM). Este es un lenguaje de bajo nivel con el que se puede programar el PIC, en la programación de PIC. Este lenguaje de programación es muy popular, en la mayoría de los foros y el mismo fabricante explica en su hoja de especificaciones cómo configurar el mismo en lenguaje ASM. Aunque no es muy amigable como el lenguaje BASIC, al programar en ASM se optimizan los recursos del PIC, ya que se facilita el trabajo de configurar registros de almacenamiento al compilar el programa ASM.

Este lenguaje de programación tiene el inconveniente de alejarse de los estándares de programación. Se compara con un if en *ASM* revisando el estado de un registro “BANDERA”.

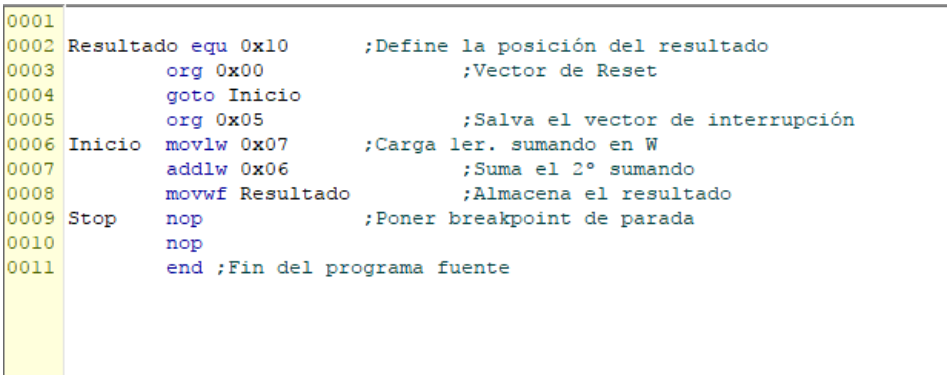
Si se desea comparar una variable con un valor predefinido se realiza una resta y si el resultado es 0 ambos valores son iguales, en cambio sí es mayor que 0 entonces la variable tiene un menor valor y por el contrario, si el valor es negativo la variable tiene un valor mayor.

Todos los programas que se utilizan para programar en *ASM* usan campos definidos, sirven para llevar un orden determinado y el compilador comprende qué hacer con cada campo:

- Campo de etiquetas
- Campo de instrucciones

- Campo de datos
- Campo de comentarios

Figura 38. **Definición de campos de programación ASM**



```
0001
0002 Resultado equ 0x10      ;Define la posición del resultado
0003         org 0x00                ;Vector de Reset
0004         goto Inicio
0005         org 0x05                ;Salva el vector de interrupción
0006 Inicio  movlw 0x07      ;Carga ler. sumando en W
0007         addlw 0x06              ;Suma el 2º sumando
0008         movwf Resultado         ;Almacena el resultado
0009 Stop    nop             ;Poner breakpoint de parada
0010         nop
0011         end ;Fin del programa fuente
```

Fuente: elaboración propia.

Las directrices son nemotécnicas que nos permiten configurar el PIC, ahorrando tiempo a la hora de compilar nuestro archivo *ASM*, el resultado de estos nemotécnicos los podemos ver en un archivo *.*lst*

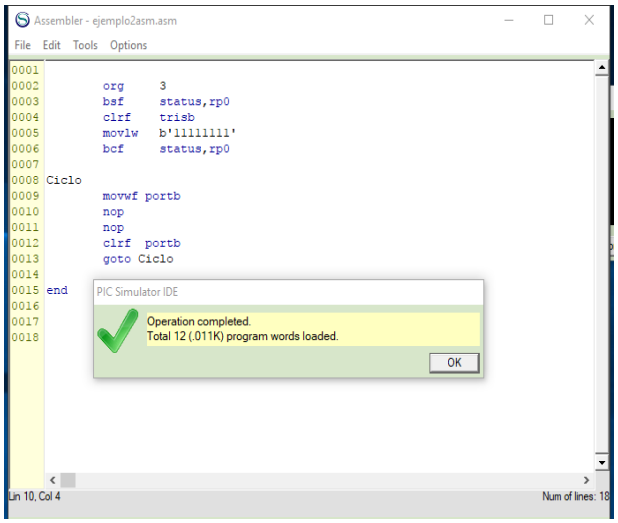
- EQU Sirve para asignar valores a las etiquetas deseadas.
- ORG (origen) indica al ensamblador dónde debe comenzar a colocar las instrucciones en la memoria de programa. La dirección de comienzo (origen) es en la posición 0, debido a que la familia de microcontroladores PIC de gama media después del encendido o *RESET* siempre ejecutan la instrucción situada en la dirección 0. Se denomina vector de reset.
- La dirección 4 es el vector de interrupción. Si se genera una interrupción el microcontrolador ejecuta la instrucción que se encuentre aquí. Es una buena práctica dejar libre la dirección 4 por si

más adelante se desea añadir capacidad de interrupción al programa. El programa salta por encima del vector de interrupción y comienza en la dirección 5.

- *INCLUDE* Incluye una lista predefinida de algunos programas que serán útiles en el programa que se ejecutará, es decir, se cargan las librerías que se utilizarán para la ejecución del programa.
- *END* indica al ensamblador el final del código fuente.

41. CONFIGURACIÓN DE PUERTO DE SALIDA PROGRAMACIÓN ASM

Figura 39. **Configuración puerto de salida en ASM**



Fuente: elaboración propia.

Línea 1 Se le da el origen del programa al PIC

Línea 2 Se direcciona al banco (0,1) donde se encuentra el registro *TRISB*

Línea 3 Se pone en 0 el valor del registro *TRISB*, con esto configuramos el puerto como puerto de salida.

Línea 4 Se le da el valor binario de 255 al registro w

Línea 5 Se regresa al banco (0,0) donde se encuentra el registro *PORTB*

Línea 6 Etiqueta de ciclo

Línea 7 Se mueve el contenido del registro w al registro *PORTB*

Línea 8 Comando de No operar (no hacer nada)

Línea 9 Comando de No operar (no hacer nada)

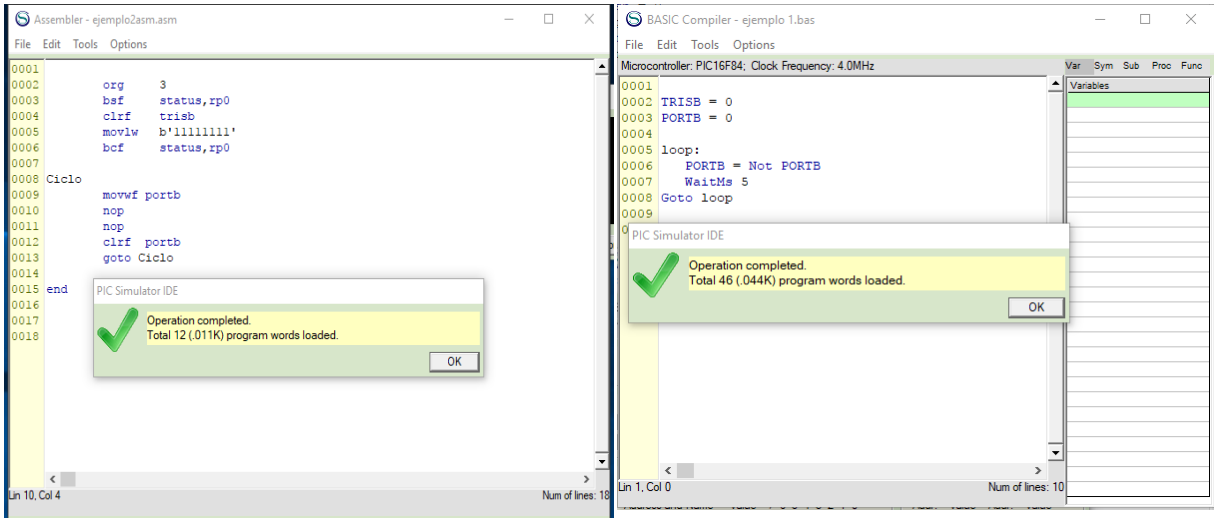
Línea 10 Se limpia el contenido del registro *PORTB*

Línea 11 Se hace un salto hacia la etiqueta Ciclo con esto realizamos un *loop* infinito

Línea 12 Fin del programa.

Es más complejo programar en ASM, sin embargo, es ventajoso al optimizar los recursos. Comparando con un *script* que realiza el mismo funcionamiento en BASIC, el programa desarrollado en el leguaje ASM ocupa mucho menos espacio en memoria.

Figura 40. **Comparación de lenguajes de programación BASIC Vs. ASM**



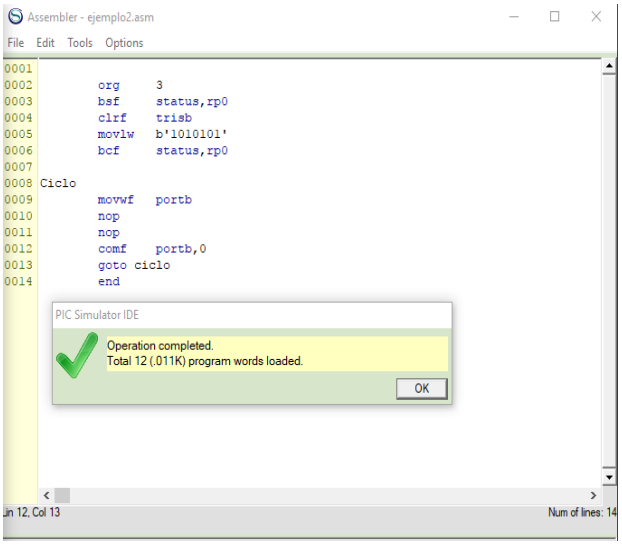
Fuente: elaboración propia.

Según la comparación, el programa desarrollado en BASIC es 4 veces más pesado que el desarrollado en ASM. Por consiguiente, el tema del lenguaje de programación que se utilizará para el desarrollo de un proyecto toma relevancia.

42. CONFIGURACIÓN DE PUERTO SALIDA EN LENGUAJE ASM

El funcionamiento del ejemplo 2 es muy similar al ejemplo 1, únicamente se agrega un nuevo nemotécnico COMF que hace el complemento del contenido del registro f bit a bit. El resultado se almacena en el registro f si d=1 y en el registro w si d=0, en este caso f no varía.

Figura 41. **Uso de nemotécnicos orientados al bit**

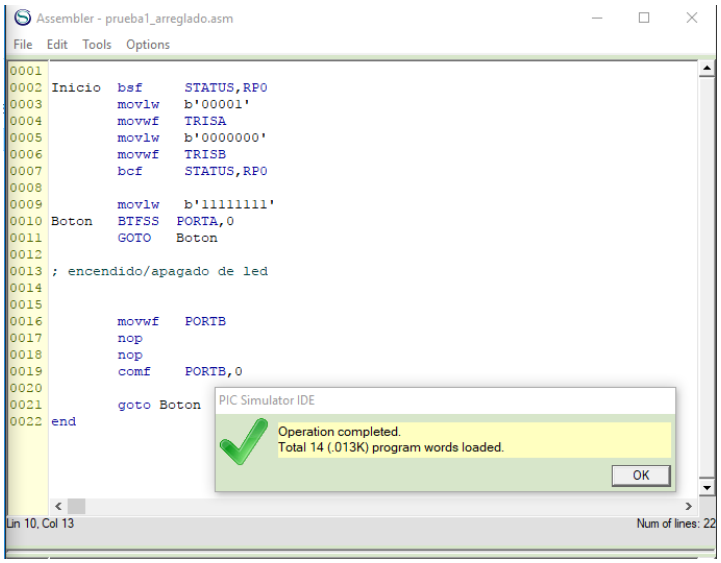


Fuente: elaboración propia.

43. CONFIGURACIÓN DE PUERTO ENTRADA/ SALIDA EN LENGUAJE ASM

El siguiente ejemplo usaremos un puerto de entrada como dispositivo de control si el pin del puerto de entrada es activado los leds empezaran a centellar.

Figura 42. **Uso de nemotécnicos orientados al bit**



Fuente: elaboración propia.

Línea 2 Se direcciona al banco (0,1) donde se encuentran los registros TRISA, TRISB.

Línea 3 Se almacena el valor 00001 en el registro W.

Línea 4 El valor w se almacena en el registro *TRISA*, con esto configuramos el pin menos significativo como puerto de entrada.

Línea 5 Se almacena el valor 00000000 en el registro W.

Línea 6 El valor del registro w se almacena en el registro *TRISB*, con esto configuramos el puerto B completo como puerto de salida.

Línea 7 Se direcciona al banco (0,0) donde se encuentran los registros *PORTA*,*PORTB*

Línea 8 Se almacena el valor 11111111 en el registro W.

Línea 9 El comando BTFSS nos sirve para comparar el valor del registro *PORTA* en la posición 0, BTFSS f,b Si el bit número b del registro f está a 1, la instrucción que sigue a ésta se ignora y se trata como un NOP (*skip*). En este caso, y solo en este caso, la instrucción BTFSS precisa dos ciclos para ejecutarse.

Línea 10 Etiqueta que nos servirá para realizar *loop*.

Línea 11 Si la respuesta del comando BTFSS es verdadero se salta la línea 11.

Línea 12 Se mueve el contenido del registro w al registro *PORTB*

Línea 13 Comando de No operar (no hacer nada)

Línea 14 Comando de No operar (no hacer nada)

Línea 15 Se almacena el complemento del valor del contendió del registro *PORTB*.

Línea 16 Se hace un salto hacia la etiqueta Botón con esto realizamos un *loop* infinito.

Línea 17 Fin del programa.

En el caso de sustituir el comando BTFSS a BTFCS se cambia la lógica del botón, mientras el botón esta oprimido no centellarán los leds.

44. CONFIGURACIÓN DE PUERTO ENTRADA/ SALIDA USO DE CONDICIONANTES EN ASM

El objetivo del programa de rutinas condicionales *IF* en ASM es hacer mover un motor DC hacia la derecha o izquierda, oprimiendo un botón para cada lado, pero si hay dos botones oprimidos o ninguno el motor no girará.

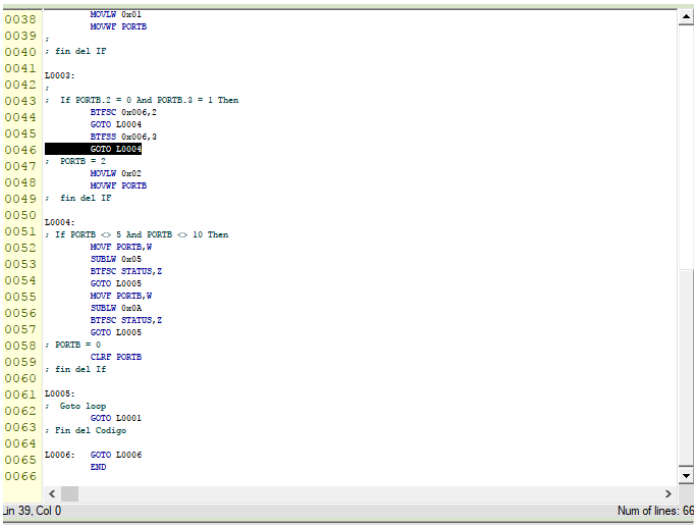
Figura 43. **Uso de condicionantes en ASM**

```
0001        ORG 0x0000
0002        BCF PCLATH,3
0003        BCF PCLATH,4
0004        GOTO L0002
0005        ORG 0x0004
0006        RETFIE
0007
0008 ; Inicio de programa
0009
0010 L0002:
0011 ;  TRISB = %00001100
0012        MOVLW 0x0C
0013        BSF STATUS,RP0
0014        MOVWF TRISB
0015 ;  TRISC = 0
0016        CLRF TRISC
0017        BCF STATUS,RP0
0018 ; loop:
0019
0020 L0001:
0021 ;
0022 ;  If PORTB.2 = 1 And PORTB.3 = 0 Then
0023        BTFSS 0x006,2
0024        GOTO L0003
0025        BTFSC 0x006,3
0026        GOTO L0003
0027 ; PORTB = 1
0028        MOVLW 0x01
0029        MOVWF PORTB
;
; fin del IF

L0003:
;
;  If PORTB.2 = 0 And PORTB.3 = 1 Then
       BTFSC 0x006,2
       GOTO L0004
       BTFSS 0x006,3
       GOTO L0004
Lin 29, Col 0                                   Num of lines: 66
```

Fuente: elaboración propia.

Figura 44. **Uso de condicionantes en ASM**



Fuente: elaboración propia.

- De la línea 1 a la 5 se inicializan los registros
- Se crea la etiqueta L0002 que es la encargada de inicializar el registro *TRISB* en configuración b'00001100.
- Se crea la etiqueta L0001 con el comando BTFSS y la BTFSS

BTFSS f, b Si el bit número b de f es nulo, la instrucción que sigue a esta se ignora y se trata como un *NOP*. En este caso, y solo en este caso, la instrucción BTFSC precisa dos ciclos para ejecutarse.

BTFSC f, b Si el bit número b de f es nulo, la instrucción que sigue a esta se ignora y se trata como un *NOP*. En este caso, y solo en este caso, la instrucción BTFSC precisa dos ciclos para ejecutarse.

Al hacer las comparaciones en las posiciones del registro PORTB 0x006,3 y 0x006,2 (0,1), se fuerza en la salida del puerto B en los valores menos significativos MOVLW 0x02 MOVWF PORTB (0,1)

- Se crea una etiqueta L0003 con los mismos comandos *BTFSC* y la *BTFSS* solo que esta vez los valores son cambiados PORTB 0x006,3 y 0x006,2 (1,0) con esto se fuerza en la salida del puerto B en los valores menos significativos MOVLW 0x01 *MOVWF PORTB* (1,0).
- Finalmente se programa si el registro *PORTB* tiene valor de 5(b'00000101) o de 10(00001010) se hace que el programa se dirija a la etiqueta L0005 que es un salto hacia la etiqueta de INICIO(L0001).

45. CONFIGURACIÓN DE TIEMPOS DE RETARDO EN LENGUAJE ASM

En este ejemplo, se utilizarán los tiempos de retardo para trabajar un tren de pulsos.

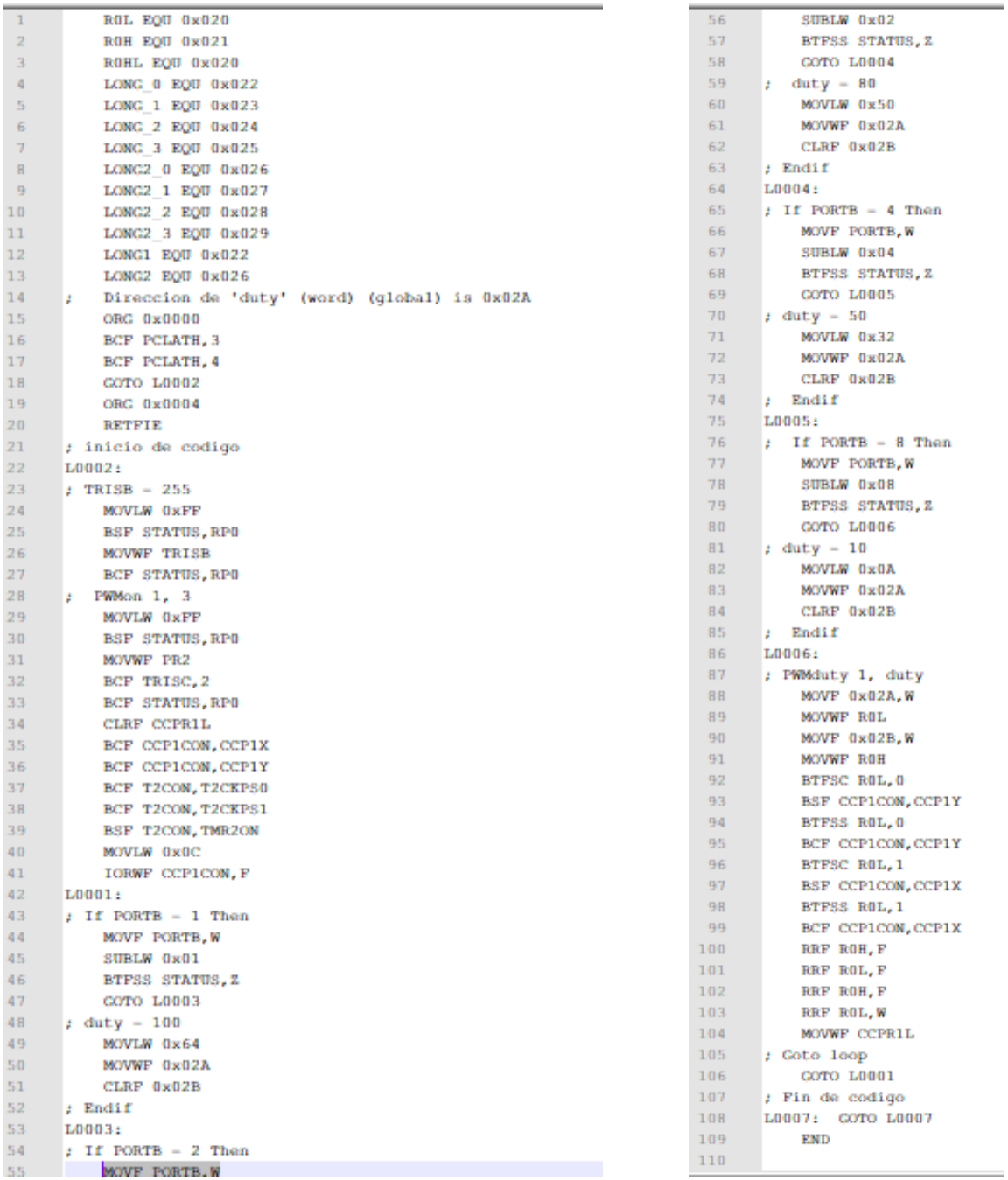
En la línea 28 se inicia la configuración de los registros de *PWM* donde los registros *CCP1CON*, *T2CON* son registros de trabajo del *PWM*. La forma de configurar estos registros se encuentra en la hoja de especificaciones del integrado.

En las líneas 46,57,68,79 se compara el valor del registro *PORTB* restando el valor del registro *PORTB* con un valor en el registro W. Si el valor es igual a 0 entonces si la bandera *STATUS*,Z se vuelve verdadera y deja pasar a la siguiente línea donde se toma el valor que controlará el tiempo que se utilizará al *PWM*.

En las líneas 49,60,71,82 se da un valor al registro W para que los registros del *PWM* puedan variar dependiendo de la selección del usuario.

De las Líneas 88 a la 104 son los movimientos de registro necesario para que el valor del registro W pueda afectar los registros del *PWM*.

Figura 45. **Programación de tiempos de retardo ASM**



```
    R0L EQU 0x020
    R0H EQU 0x021
    R0HL EQU 0x020
    LONG_0 EQU 0x022
    LONG_1 EQU 0x023
    LONG_2 EQU 0x024
    LONG_3 EQU 0x025
    LONG2_0 EQU 0x026
    LONG2_1 EQU 0x027
    LONG2_2 EQU 0x028
    LONG2_3 EQU 0x029
    LONG1 EQU 0x022
    LONG2 EQU 0x026
;   Direccion de 'duty' (word) (global) is 0x02A
    ORG 0x0000
    BCF PCLATH,3
    BCF PCLATH,4
    GOTO L0002
    ORG 0x0004
    RETFIE
; inicio de codigo
L0002:
; TRISB = 255
    MOVLW 0xFF
    BSF STATUS,RP0
    MOVWF TRISB
    BCF STATUS,RP0
;  PWMon 1, 3
    MOVLW 0xFF
    BSF STATUS,RP0
    MOVWF PR2
    BCF TRISC,2
    BCF STATUS,RP0
    CLRF CCPR1L
    BCF CCP1CON,CCP1X
    BCF CCP1CON,CCP1Y
    BCF T2CON,T2CKPS0
    BCF T2CON,T2CKPS1
    BSF T2CON,TMR2ON
    MOVLW 0x0C
    IORWF CCP1CON,F
L0001:
; If PORTB = 1 Then
    MOVF PORTB,W
    SUBLW 0x01
    BTFSS STATUS,Z
    GOTO L0003
; duty = 100
    MOVLW 0x64
    MOVWF 0x02A
    CLRF 0x02B
; Endif
L0003:
; If PORTB = 2 Then
    MOVF PORTB,W
    SUBLW 0x02
    BTFSS STATUS,Z
    GOTO L0004
;  duty = 80
    MOVLW 0x50
    MOVWF 0x02A
    CLRF 0x02B
; Endif
L0004:
; If PORTB = 4 Then
    MOVF PORTB,W
    SUBLW 0x04
    BTFSS STATUS,Z
    GOTO L0005
; duty = 50
    MOVLW 0x32
    MOVWF 0x02A
    CLRF 0x02B
;  Endif
L0005:
;  If PORTB = 8 Then
    MOVF PORTB,W
    SUBLW 0x08
    BTFSS STATUS,Z
    GOTO L0006
; duty = 10
    MOVLW 0x0A
    MOVWF 0x02A
    CLRF 0x02B
;  Endif
L0006:
; PWMduty 1, duty
    MOVF 0x02A,W
    MOVWF R0L
    MOVF 0x02B,W
    MOVWF R0H
    BTFSC R0L,0
    BSF CCP1CON,CCP1Y
    BTFSS R0L,0
    BCF CCP1CON,CCP1Y
    BTFSC R0L,1
    BSF CCP1CON,CCP1X
    BTFSS R0L,1
    BCF CCP1CON,CCP1X
    RRF R0H,F
    RRF R0L,F
    RRF R0H,F
    RRF R0L,W
    MOVWF CCPR1L
; Goto loop
    GOTO L0001
; Fin de codigo
L0007:  GOTO L0007
    END
```

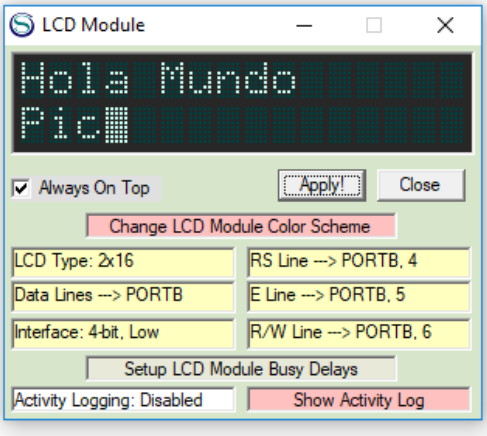
Fuente: elaboración propia.

46. CONFIGURACIÓN DE LIBRERÍAS EN LENGUAJE ASM

En el siguiente ejemplo, se identifica la necesidad de adaptar librerías predefinidas. De esta manera el esfuerzo se centra en el análisis de información, que es lo importante en el programa.

Figura 46. **Programación de librerías ASM**

```
    R0L EQU 0x020
    R0H EQU 0x021
    R3L EQU 0x026
    R3H EQU 0x027
    R4L EQU 0x028
    R4H EQU 0x029
    R0HL EQU 0x020
    R3HL EQU 0x026
    R4HL EQU 0x028
    LONG_0 EQU 0x02A
    LONG_1 EQU 0x02B
    LONG_2 EQU 0x02C
    LONG_3 EQU 0x02D
    LONG2_0 EQU 0x02E
    LONG2_1 EQU 0x02F
    LONG2_2 EQU 0x030
    LONG2_3 EQU 0x031
    LONG1 EQU 0x02A
    LONG2 EQU 0x02E
    ORG 0x0000
    BCF PCLATH,3
    BCF PCLATH,4
    GOTO L0002
    ORG 0x0004
    RETFIE
; User code start
L0002:
; 1: Define LCD_LINES = 2
; 2: Define LCD_CHARS = 16
; 3: Define LCD_BITS = 4
; 4: Define LCD_DREG = PORTB
; 5: Define LCD_DBIT = 0
; 6: Define LCD_RSREG = PORTB
; 7: Define LCD_RSBIT = 4
; 8: Define LCD_EREG = PORTB
; 9: Define LCD_EBIT = 5
; 10: Define LCD_RWREG = PORTB
; 11: Define LCD_RWBIT = 6
; 12:
; 13: AllDigital
    MOVLW 0x06
    BSF STATUS,RP0
    MOVWF ADCON1
```



Fuente: elaboración propia.

Figura 47. **Programación de Librerías ASM**

```
42      BSF STATUS,RP0
43      MOVWF ADCON1
44      MOVLW 0x07
45      MOVWF CMCON
46      BCF STATUS,RP0
47  ; 14: Lcdinit 3
48      BCF PORTB,5
49      BCF PORTB,4
50      BCF PORTB,6
51      BSF STATUS,RP0
52      BCF TRISB,5
53      BCF TRISB,4
54      BCF TRISB,6
55      MOVLW 0xF0
56      ANDWF TRISB,F
57      BCF STATUS,RP0
58      MOVLW 0x64
59      MOVWF R0L
60      CLRF R0H
61      CALL W001
62      MOVLW 0x33
63      CALL LC00
64      MOVLW 0xE6
65      MOVWF R4L
66      MOVLW 0x03
67      MOVWF R4H
68      CALL DL02
69      MOVLW 0x33
70      CALL LC00
71      MOVLW 0x41
72      MOVWF R4L
73      CALL DL01
74      MOVLW 0x33
75      CALL LC00
76      MOVLW 0x41
77      MOVWF R4L
78      CALL DL01
79      MOVLW 0x22
80      CALL LC00
81      MOVLW 0x41
82      MOVWF R4L
83      CALL DL01
84      MOVLW 0x28
85      CALL LC02
86      MOVLW 0x0F
87      CALL LC02
88      MOVLW 0x01
89      CALL LC02
90  ; 15: loop:
91  L0001:
92  ; 16: Lcdcmdout LcdClear
93      MOVLW 0x01
94      CALL LC02
95  ; 17: Lcdcmdout LcdLine1Home
96      MOVLW 0x80
```

```
96      MOVLW 0x80
97      CALL LC02
98  ; 18: Lcdout "Hola Mundo"
99      MOVLW 0x48
100     CALL LC01
101     MOVLW 0x6F
102     CALL LC01
103     MOVLW 0x6C
104     CALL LC01
105     MOVLW 0x61
106     CALL LC01
107     MOVLW 0x20
108     CALL LC01
109     MOVLW 0x4D
110     CALL LC01
111     MOVLW 0x75
112     CALL LC01
113     MOVLW 0x6E
114     CALL LC01
115     MOVLW 0x64
116     CALL LC01
117     MOVLW 0x6F
118     CALL LC01
119 ; 19: Lcdcmdout LcdLine2Home
120     MOVLW 0xC0
121     CALL LC02
122 ; 20: Lcdout "Pic"
123     MOVLW 0x50
124     CALL LC01
125     MOVLW 0x69
126     CALL LC01
127     MOVLW 0x63
128     CALL LC01
129 ; 21: WaitMs 1000
130     MOVLW 0xE8
131     MOVWF R0L
132     MOVLW 0x03
133     MOVWF R0H
134     CALL W001
135 ; 22: Lcdcmdout LcdLine1Clear
136     MOVLW 0x80
137     CALL LC02
138     MOVLW 0x10
139     MOVWF R0L
140 L0003:  MOVLW 0x20
141     CALL LC01
142     DECFSZ R0L,F
143     GOTO L0003
144     MOVLW 0x80
145     CALL LC02
146 ; 23: Lcdcmdout LcdLine2Clear
147     MOVLW 0xC0
148     CALL LC02
149     MOVLW 0x10
150     MOVWF R0L
```

Fuente: elaboración propia.

Figura 48. **Programación de Librerías ASM**

```
150     MOVWF R0L
151 L0004:  MOVLW 0x20
152     CALL LC01
153     DECFSZ R0L,F
154     GOTO L0004
155     MOVLW 0xC0
156     CALL LC02
157 ; 24: Lcdcmdout LcdLine1Pos(1)
158     MOVLW 0x80
159     CALL LC02
160 ; 25: Lcdout "Linea 1"
161     MOVLW 0x4C
162     CALL LC01
163     MOVLW 0x69
164     CALL LC01
165     MOVLW 0x6E
166     CALL LC01
167     MOVLW 0x65
168     CALL LC01
169     MOVLW 0x61
170     CALL LC01
171     MOVLW 0x20
172     CALL LC01
173     MOVLW 0x31
174     CALL LC01
175 ; 26: Lcdcmdout LcdLine2Pos(2)
176     MOVLW 0xC1
177     CALL LC02
178 ; 27: Lcdout "Linea 2"
179     MOVLW 0x4C
180     CALL LC01
181     MOVLW 0x69
182     CALL LC01
183     MOVLW 0x6E
184     CALL LC01
185     MOVLW 0x65
186     CALL LC01
187     MOVLW 0x61
188     CALL LC01
189     MOVLW 0x20
190     CALL LC01
191     MOVLW 0x32
192     CALL LC01
193 ; 28: WaitMs 1000
194     MOVLW 0xE8
195     MOVWF R0L
196     MOVLW 0x03
197     MOVWF R0H
198     CALL W001
199 ; 29: Goto loop
200     GOTO L0001
201 ; End of user code
202 L0005:  GOTO L0005
203 ;
204 ;
```

Fuente: elaboración propia

Figura 49. **Programación de librerías ASM**

```
;
; Delay Routine Byte
; minimal routine execution time: 8µs
; routine execution time step: 3µs
; maximal routine execution time: 770µs
DL01:
    DECFSZ R4L,F
    GOTO DL01
    RETURN
; Delay Routine Word
; minimal routine execution time: 15µs
; routine execution time step: 10µs
; maximal routine execution time: 655365µs
DL02:
    MOVLW 0x01
    SUBWF R4L,F
    CLRW
    BTFSS STATUS,C
    ADDLW 0x01
    SUBWF R4H,F
    BTFSS STATUS,C
    RETURN
    GOTO DL02
; Waitms Routine
W001:   MOVLW 0x01
    SUBWF R0L,F
    CLRW
    BTFSS STATUS,C
    ADDLW 0x01
    SUBWF R0H,F
    BTFSS STATUS,C
    RETURN
    MOVLW 0x61
    MOVWF R4L
    MOVLW 0x00
    MOVWF R4H
    CALL DL02
    GOTO W001
; Lcdout Routine
LC01:
    MOVWF R3L
    BSF PORTB,4
    BCF PORTB,6
    MOVF PORTB,W
    MOVWF R3H
    SWAPF R3L,F
    CALL LCX5
    MOVWF PORTB
    CALL LCX1
    MOVF PORTB,W
    MOVWF R3H
    SWAPF R3L,F
    CALL LCX5
    MOVWF PORTB
    CALL LCX1
    MOVLW 0x1F
    MOVWF R4L
    CALL DL01
    RETURN
LCX5:
    MOVLW 0xF0
    ANDWF R3H,F
    MOVF R3L,W
    ANDLW 0x0F
    IORWF R3H,W
    RETURN
LCX1:
    BSF PORTB,5
    NOP
    BCF PORTB,5
    NOP
    RETURN
; Lcdinit Routine
LC00:
    BCF PORTB,4
    BCF PORTB,6
    MOVWF R3L
    MOVF PORTB,W
    MOVWF R3H
    CALL LCX5
    MOVWF PORTB
    CALL LCX1
    RETURN
; Lcdcmdout Routine
LC02:
    MOVWF R3L
    BCF PORTB,4
    BCF PORTB,6
    MOVF PORTB,W
    MOVWF R3H
    SWAPF R3L,F
    CALL LCX5
    MOVWF PORTB
    CALL LCX1
    MOVF PORTB,W
    MOVWF R3H
    SWAPF R3L,F
    CALL LCX5
    MOVWF PORTB
    CALL LCX1
    MOVLW 0xF2
    MOVWF R4L
    MOVLW 0x01
    MOVWF R4H
    CALL DL02
    RETURN

; End of listing
    END
```

Fuente: elaboración propia.

Son 315 líneas para mostrar 4 mensajes en la pantalla *LCD* donde las letras por mostrar únicamente ocupan de la línea 99 a la 108 para el “HOLA MUNDO” “PIC”, donde cada valor que se mueve al registro *W* es el equivalente a su valor de la letra que se desea mostrar, y la rutina *CALL* es el llamado a la librería encargada de trabajar la LCD.

CONCLUSIONES

1. El presente trabajo de graduación es un Manual para el laboratorio de Microcontroladores PIC de la Escuela Ingeniería Mecánica Eléctrica, con el cual se explica paso a paso cómo se debe de trabajar un microprocesador.

2. Es importante que el estudiante, además de saber programar un microprocesador, sepa optimizar sus recursos por tal motivo. Los ejemplos se explican en dos lenguajes de programación distintos, así el estudiante puede elegir entre dos programas según sus ventajas y desventajas.

3. Tomar en cuenta que la mayoría de los foros de programación están enfocados en lenguaje de programación ASM por lo tanto es importante que el estudiante esté familiarizado con este lenguaje de programación.

4. Se explica cómo se debe seleccionar un microprocesador ya que entre el estudiantado surgen tendencias de uso de microprocesadores bajo el slogan “Si a él le funcionó con este micro a mí también”. La falta de información propicia la compra de microprocesadores sobre calificados para el trabajo que se debe realizar y gastos económicos innecesarios.

RECOMENDACIONES

1. Nunca quedarse con la duda en estos tiempos, el internet es una gran herramienta de trabajo y tomando en cuenta la diversidad de la misma, podemos tomar diferentes puntos de vista para una misma solución.

2. Estar seguro de los voltajes de alimentación de un Microprocesador ya que estos elementos son muy sensibles a los problemas de energía.

3. Antes de energizar los circuitos se debe verificar que no haya capacitores de voltajes grandes cargados ya que funcionan como pequeñas baterías que aumentan el voltaje del circuito.

4. Siempre se debe armar el circuito de Reset para evitar la supresión de energía en el circuito.

5. Una manera sencilla de conseguir muestras gratis de microprocesadores es crear una cuenta en la página oficial de microchip con un correo académico, las muestras son limitadas y no todo el catalogo está disponible para esta opción, la solicitud de las muestras son periódicas, pero es una buena manera de conseguir microprocesadores que no están al alcance comercial normalmente.

BIBLIOGRAFÍA

1. Microchip Technology Inc. Data Sheet PIC 16F877A. *Catálogo de hojas de datos [en línea]*. http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf. [Consulta: 8 octubre 2017].

2. STMicroelectronics GROUP OF COMPANIES Data Sheet L298 H Bridge *catálogo de hojas de datos [en línea].* https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf . [Consulta: 8 octubre 2017].

3. Hitachi, Ltd. Semiconductor & Integrated CircuitsData Sheet pantalla LCD 16x2 HD44780. *Catálogo de hojas de datos [en línea].* https://www.sparkfun.com/datasheets/LCD/HD44780.pdf. [Consulta: 20 septiembre 2017].

APÉNDICES

Apéndice 1. **Características básicas**

CARACTERÍSTICAS	16F877
Frecuencia máxima	DX-20MHz
Memoria de programa flash palabra de 14 bits	8KB
Posiciones RAM de datos	368
Posiciones EEPROM de datos	256
Puertos E/S	A,B,C,D,E
Número de pines	40
Interrupciones	14
Timers	3
Módulos CCP	2
Comunicaciones Serie	MSSP, USART
Comunicaciones paralelo	PSP
Líneas de entrada de CAD de 10 bits	8
Juego de instrucciones	35 instrucciones
Longitud de la instrucción	14 bits
Arquitectura	Harvard
CPU	Risc
Canales Pwm	2

Fuente: elaboración propia.

Apéndice 2. **Características básica por pines**

Nomenclatura	Descripción	Nomenclatura	Descripción
AN0	Entrada Analógica 0	**RD6**	Registro D Digital I/O pin 6
AN1	Entrada Analógica 1	**RD7**	Registro D Digital I/O pin 7
AN2	ntrada Analógica 2 ó Voltaje de referenc	**RE0**	Registro E Digital I/O pin 0
	negativo	**RE1**	Registro E Digital I/O pin 1
AN3	ntrada Analógica 3 ó Voltaje de referenc	**RE2**	Registro E Digital I/O pin 2
	positivo	**RX**	Puerto *USART* TX en asíncrono
AN4	Entrada Analógica 4	**SCK**	*Synchronous serial clock input*
AN5	Entrada Analógica 6	**SCL**	Sincronización puerto serial
AN6	Entrada Analógica 7	**SDA**	Puerto Datos IC
AN7	Entrada Analógica 8	**SDI**	Datos de entrada SPI
C1OUT	Salida de comparador 1	**SDO**	Datos de salida SPI
C2OUT	Salida de comparador 2	**SS**	*SPI Slave Select input.*
CCP1	Salida PWD1	**T0CKI**	*Timer0 external clock input*
CCP2	Salida PWD2	**T1CKI**	Entrada Reloj Externo 1
CK	Puerto USART Reloj síncrono	**T1OSI**	Entrada *Timer1* Oscilador
CLKI	asociación de relojes interno/externo	**T1OSO**	Salida Timer1 Oscilador
CLKO	entrada Oscilador RC	**TX**	Puerto *USART* RX en asíncrono
CS	Selección y control puerto Esclavo	**VDD**	Voltaje positivo
CVREF	Comparador de voltaje de referencia	**VPP**	*Programming Voltage Input*.
DT	Puerto USART Datos síncrono	**VREF-**	Comparador de voltaje de referencia
INT	Interrupción Externa	**VREF+**	Comparador de voltaje de referencia
MCLR	*Master Clear Reset*	**VSS**	Tierra
NC	No Conectar	**WR**	Escritura y control puerto Esclavo
OSC1	entrada oscilador de cristal		
OSC2	entrada oscilador de cristal con		
	conexión resonante		
PGC	In-circuit debugger and ICSP		
	programming clock.		
PGD	-circuit debugger and ICSP programming		
	clock.		
PGM	*Low-voltage (single-supply) ICSP*		
	programming enable pin.		
PSP0	Puerto Paralelo 0		
PSP1	Puerto Paralelo 1		
PSP2	Puerto Paralelo 2		
PSP3	Puerto Paralelo 3		
PSP4	Puerto Paralelo 4		
PSP5	Puerto Paralelo 5		
PSP6	Puerto Paralelo 6		
PSP7	Puerto Paralelo 7		
RA0	Registro A Digital I/O pin 0		
RA1	Registro A Digital I/O pin 1		
RA2	Registro A Digital I/O pin 2		
RA3	Registro A Digital I/O pin 3		
RA4	Registro A Digital I/O pin 4		
RA5	Registro A Digital I/O pin 5		
RB0	Registro B Digital I/O pin 0		
RB1	Registro B Digital I/O pin 1		
RB2	Registro B Digital I/O pin 2		
RB3	Registro B Digital I/O pin 3		
RB4	Registro B Digital I/O pin 4		
RB5	Registro B Digital I/O pin 5		
RB6	Registro B Digital I/O pin 6		
RB7	Registro B Digital I/O pin 7		
RC0	Registro C Digital I/O pin 0		
RC1	Registro C Digital I/O pin 1		
RC2	Registro C Digital I/O pin 2		
RC3	Registro C Digital I/O pin 3		
RC4	Registro C Digital I/O pin 4		
RC5	Registro C Digital I/O pin 5		
RC6	Registro C Digital I/O pin 6		
RC7	Registro C Digital I/O pin 7		
RD	Lectura y control puerto Esclavo		
RD0	Registro D Digital I/O pin 0		
RD1	Registro D Digital I/O pin 1		
RD2	Registro D Digital I/O pin 2		
RD3	Registro D Digital I/O pin 3		
RD4	Registro D Digital I/O pin 4		
RD5	Registro D Digital I/O pin 5		

Fuente: elaboración propia.

Apéndice 3. **Tabla de conversión numérica dec,hex, oct**

Decimal	Binario	Hex	Decimal	Binario	Hex	Decimal	Binario	Hex	Decimal	Binario	Hex
0	0	0	32	100000	20	64	1000000	40	96	1100000	60
1	1	1	33	100001	21	65	1000001	41	97	1100001	61
2	10	2	34	100010	22	66	1000010	42	98	1100010	62
3	11	3	35	100011	23	67	1000011	43	99	1100011	63
4	100	4	36	100100	24	68	1000100	44	100	1100100	64
5	101	5	37	100101	25	69	1000101	45	101	1100101	65
6	110	6	38	100110	26	70	1000110	46	102	1100110	66
7	111	7	39	100111	27	71	1000111	47	103	1100111	67
8	1000	8	40	101000	28	72	1001000	48	104	1101000	68
9	1001	9	41	101001	29	73	1001001	49	105	1101001	69
10	1010	A	42	101010	2A	74	1001010	4A	106	1101010	6A
11	1011	B	43	101011	2B	75	1001011	4B	107	1101011	6B
12	1100	C	44	101100	2C	76	1001100	4C	108	1101100	6C
13	1101	D	45	101101	2D	77	1001101	4D	109	1101101	6D
14	1110	E	46	101110	2E	78	1001110	4E	110	1101110	6E
15	1111	F	47	101111	2F	79	1001111	4F	111	1101111	6F
16	10000	10	48	110000	30	80	1010000	50	112	1110000	70
17	10001	11	49	110001	31	81	1010001	51	113	1110001	71
18	10010	12	50	110010	32	82	1010010	52	114	1110010	72
19	10011	13	51	110011	33	83	1010011	53	115	1110011	73
20	10100	14	52	110100	34	84	1010100	54	116	1110100	74
21	10101	15	53	110101	35	85	1010101	55	117	1110101	75
22	10110	16	54	110110	36	86	1010110	56	118	1110110	76
23	10111	17	55	110111	37	87	1010111	57	119	1110111	77
24	11000	18	56	111000	38	88	1011000	58	120	1111000	78
25	11001	19	57	111001	39	89	1011001	59	121	1111001	79
26	11010	1A	58	111010	3A	90	1011010	5A	122	1111010	7A
27	11011	1B	59	111011	3B	91	1011011	5B	123	1111011	7B
28	11100	1C	60	111100	3C	92	1011100	5C	124	1111100	7C
29	11101	1D	61	111101	3D	93	1011101	5D	125	1111101	7D
30	11110	1E	62	111110	3E	94	1011110	5E	126	1111110	7E
31	11111	1F	63	111111	3F	95	1011111	5F	127	1111111	7F
128	10000000	80	160	10100000	A0	192	11000000	C0	224	11100000	E0
129	10000001	81	161	10100001	A1	193	11000001	C1	225	11100001	E1
130	10000010	82	162	10100010	A2	194	11000010	C2	226	11100010	E2
131	10000011	83	163	10100011	A3	195	11000011	C3	227	11100011	E3
132	10000100	84	164	10100100	A4	196	11000100	C4	228	11100100	E4
133	10000101	85	165	10100101	A5	197	11000101	C5	229	11100101	E5
134	10000110	86	166	10100110	A6	198	11000110	C6	230	11100110	E6
135	10000111	87	167	10100111	A7	199	11000111	C7	231	11100111	E7
136	10001000	88	168	10101000	A8	200	11001000	C8	232	11101000	E8
137	10001001	89	169	10101001	A9	201	11001001	C9	233	11101001	E9
138	10001010	8A	170	10101010	AA	202	11001010	CA	234	11101010	EA
139	10001011	8B	171	10101011	AB	203	11001011	CB	235	11101011	EB
140	10001100	8C	172	10101100	AC	204	11001100	CC	236	11101100	EC
141	10001101	8D	173	10101101	AD	205	11001101	CD	237	11101101	ED
142	10001110	8E	174	10101110	AE	206	11001110	CE	238	11101110	EE
143	10001111	8F	175	10101111	AF	207	11001111	CF	239	11101111	EF
144	10010000	90	176	10110000	B0	208	11010000	D0	240	11110000	F0
145	10010001	91	177	10110001	B1	209	11010001	D1	241	11110001	F1
146	10010010	92	178	10110010	B2	210	11010010	D2	242	11110010	F2
147	10010011	93	179	10110011	B3	211	11010011	D3	243	11110011	F3
148	10010100	94	180	10110100	B4	212	11010100	D4	244	11110100	F4
149	10010101	95	181	10110101	B5	213	11010101	D5	245	11110101	F5
150	10010110	96	182	10110110	B6	214	11010110	D6	246	11110110	F6
151	10010111	97	183	10110111	B7	215	11010111	D7	247	11110111	F7
152	10011000	98	184	10111000	B8	216	11011000	D8	248	11111000	F8
153	10011001	99	185	10111001	B9	217	11011001	D9	249	11111001	F9
154	10011010	9A	186	10111010	BA	218	11011010	DA	250	11111010	FA
155	10011011	9B	187	10111011	BB	219	11011011	DB	251	11111011	FB
156	10011100	9C	188	10111100	BC	220	11011100	DC	252	11111100	FC
157	10011101	9D	189	10111101	BD	221	11011101	DD	253	11111101	FD
158	10011110	9E	190	10111110	BE	222	11011110	DE	254	11111110	FE
159	10011111	9F	191	10111111	BF	223	11011111	DF	255	11111111	FF

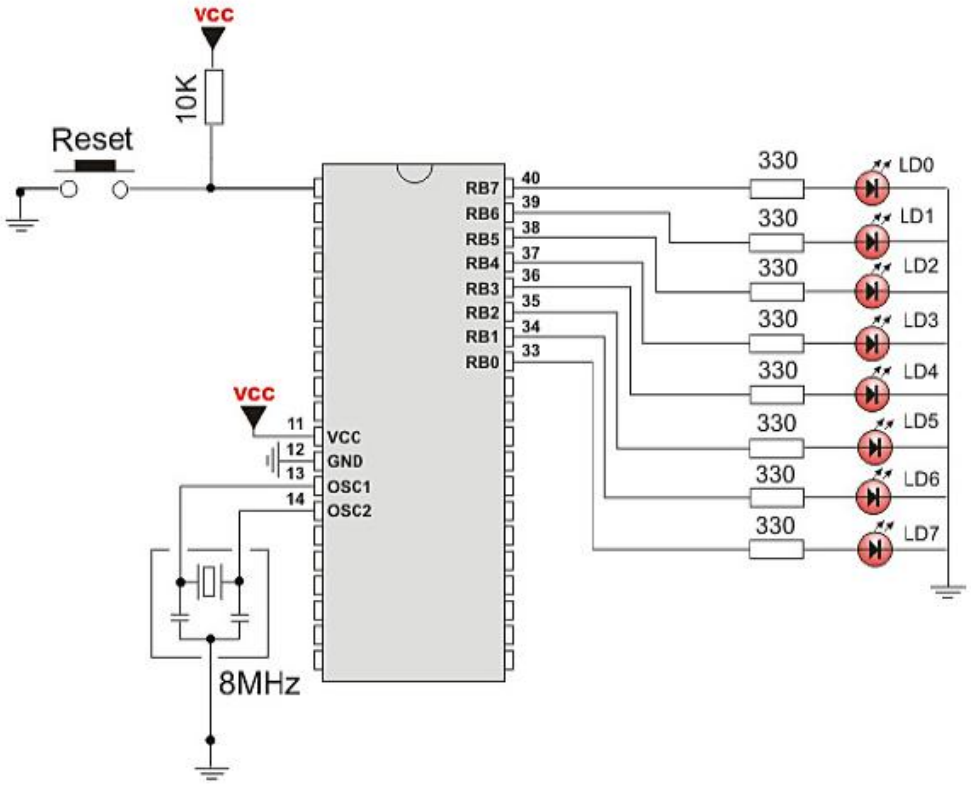
Fuente: elaboración propia.

Apéndice 4. **Valores por puerto para display**

Valor a desplegar	ánodo común	Cátodo común
0	192	63
1	249	6
2	164	91
3	176	79
4	153	102
5	146	109
6	130	125
7	248	7
8	128	127
9	152	103
a	136	119
b	131	124
c	198	57
d	161	94
e	134	121
f	142	113
g	144	111
h	139	116
i	249	6
j	225	30
l	199	56
m	43	212
n	171	84
o	163	92
p	140	115
q	152	103
r	175	80
s	146	109
t	135	120
u	193	62
v	65	190
y	25	230
z	36	219

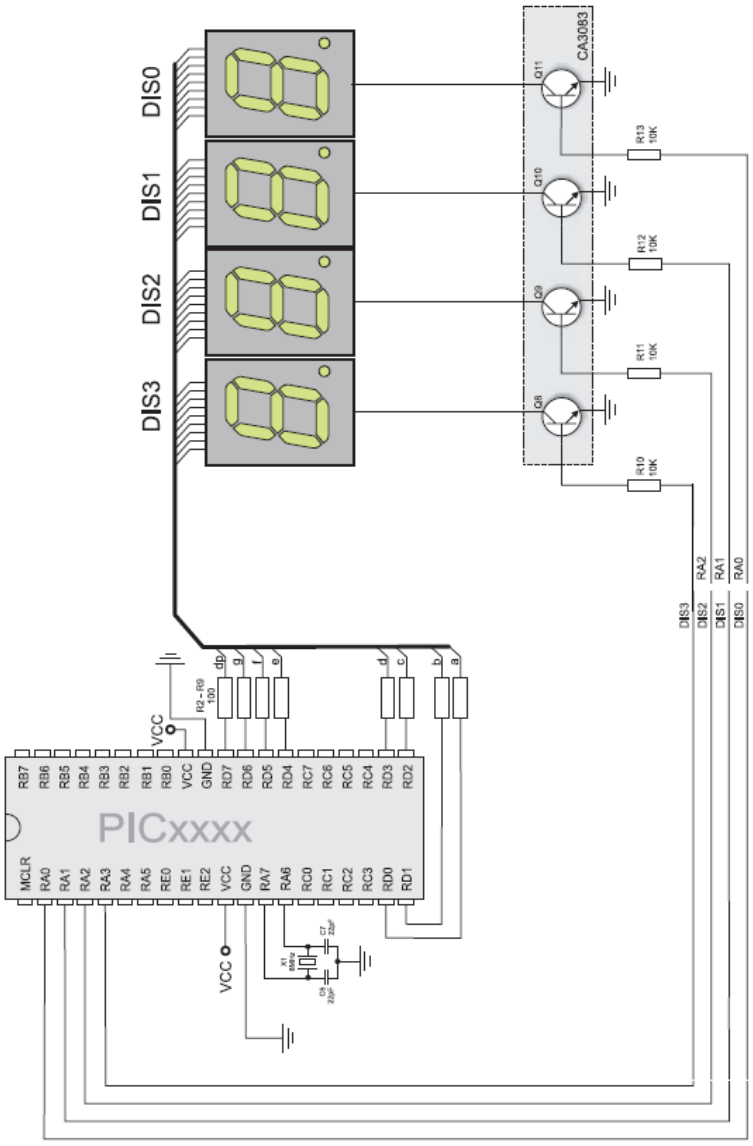
Fuente: elaboración propia.

Apéndice 5. **Conexión básica de un pic**



Fuente: elaboración propia.

Apéndice 6. **Conexión circuito multiplexado display**



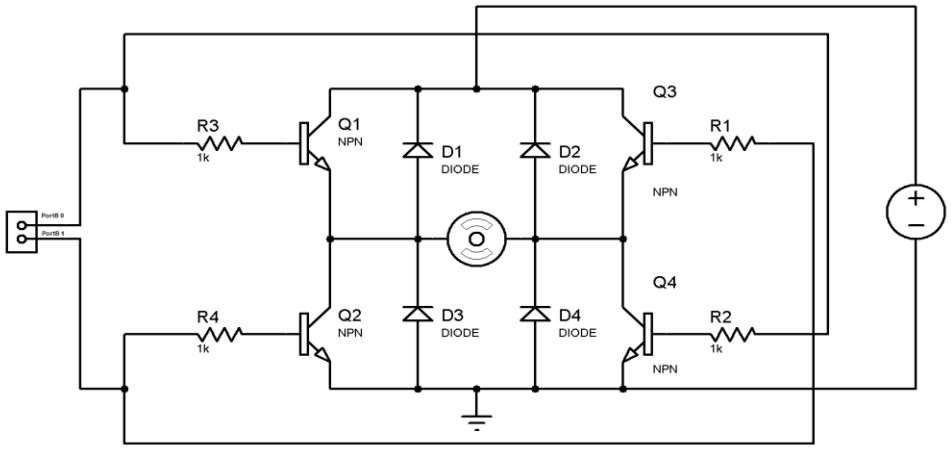
Fuente: elaboración propia.

Apéndice 7. **Valores por puerto para motor de giro**

PortB		
0	**1**	**Giro**
0	0	No Giro
0	1	Giro Derecho
1	0	Giro Izquierdo

Fuente: elaboración propia.

Apéndice 8. **Circuito básico giro motor DC con puente H**



Fuente: elaboración propia.

Apéndice 9. **Configuración de motores de paso**

Valores para motor de paso en modo velocidad

Velocidad					
Fase 1	Fase 2	Fase 3	Fase 4	Decimal	hexadecimal
1	0	0	0	8	8
0	1	0	0	4	4
0	0	1	0	2	2
0	0	0	1	1	1

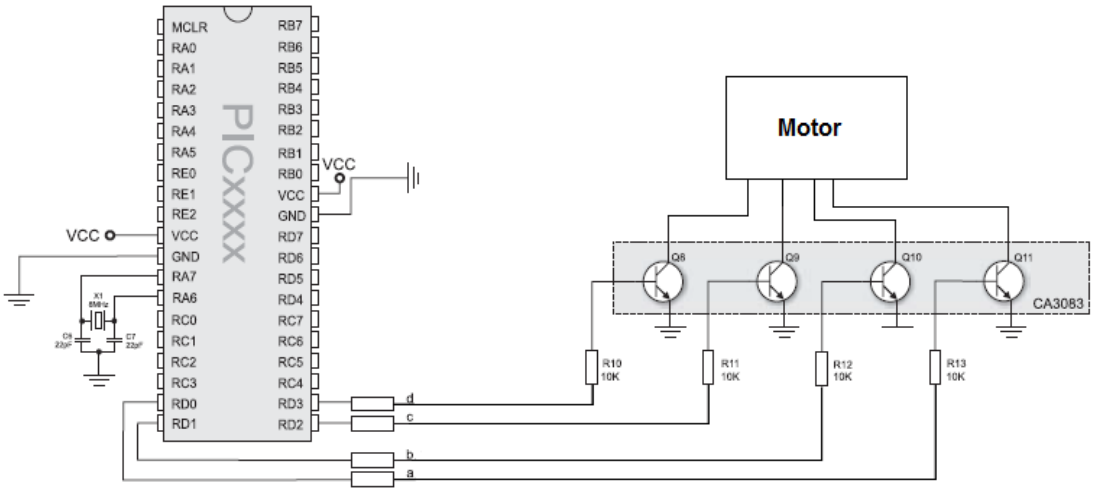
Fuente: elaboración propia.

Apéndice 10. **Valores para motor de paso en modo torque**

Torque					
Fase 1	Fase 2	Fase 3	Fase 4	decimal	Hexadecimal
1	1	0	0	12	C
0	1	1	0	6	6
0	0	1	1	3	3
1	0	0	1	9	9

Fuente: elaboración propia.

Apéndice 11. **Circuito básico motor de paso**



Fuente: elaboración propia.

Apéndice 12. **Resolucion ancho de pulso PWM**

$$\text{Resolucion} = \frac{\log\left(\frac{\text{Fosc}}{\text{Fpwm}}\right)}{\log(2)}$$

PWM Periodo = [(PR2) + 1]*4* TOSC*(TMR2 Valor Preescala)

PWM Duty Cycle =(CCPR1L:CCP1CON)*TOSC*(TMR2 Valor Preescala)

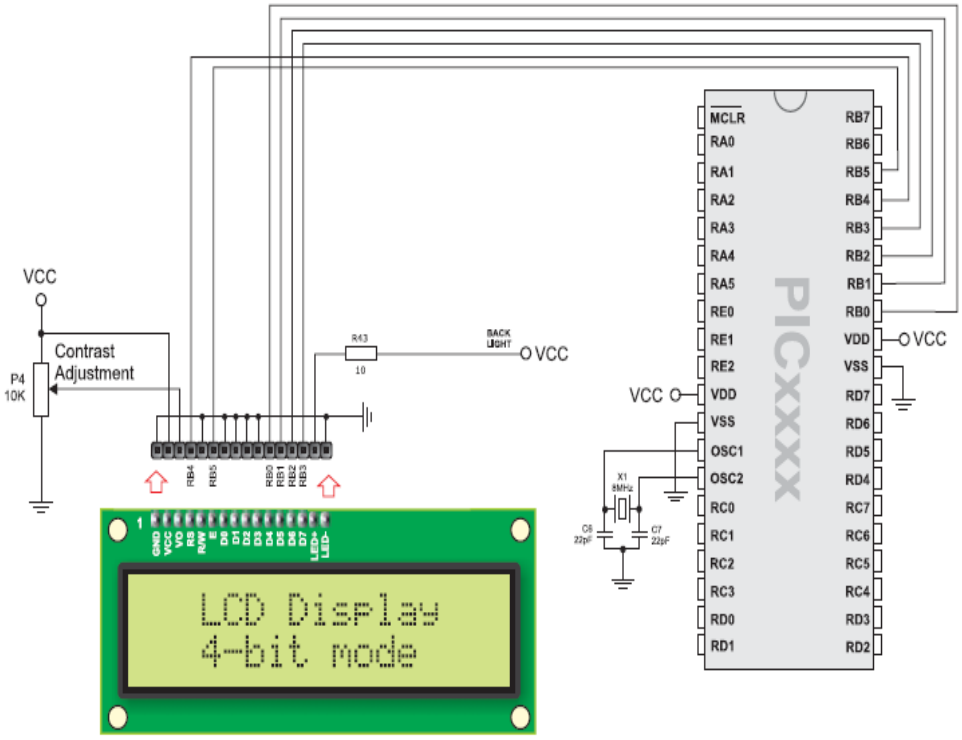
Fuente: elaboración propia.

Apéndice 13. **Resolucion ancho de pulso PWM**

Ejemplo de frecuencia y resolución con un reloj 20 MHz						
PWM Frecuencia	1.22kHz	4.88kHz	19.53kHz	78.12kHz	156.3kHz	208.3kHz
Tiempo en Preescala	16	4	1	1	1	1
PR2 Valor	0xFFh	0xFFh	0xFFh	0x3Fh	0x1Fh	0x17h
Maxima Resolución (bits)	10	10	10	8	7	5.5

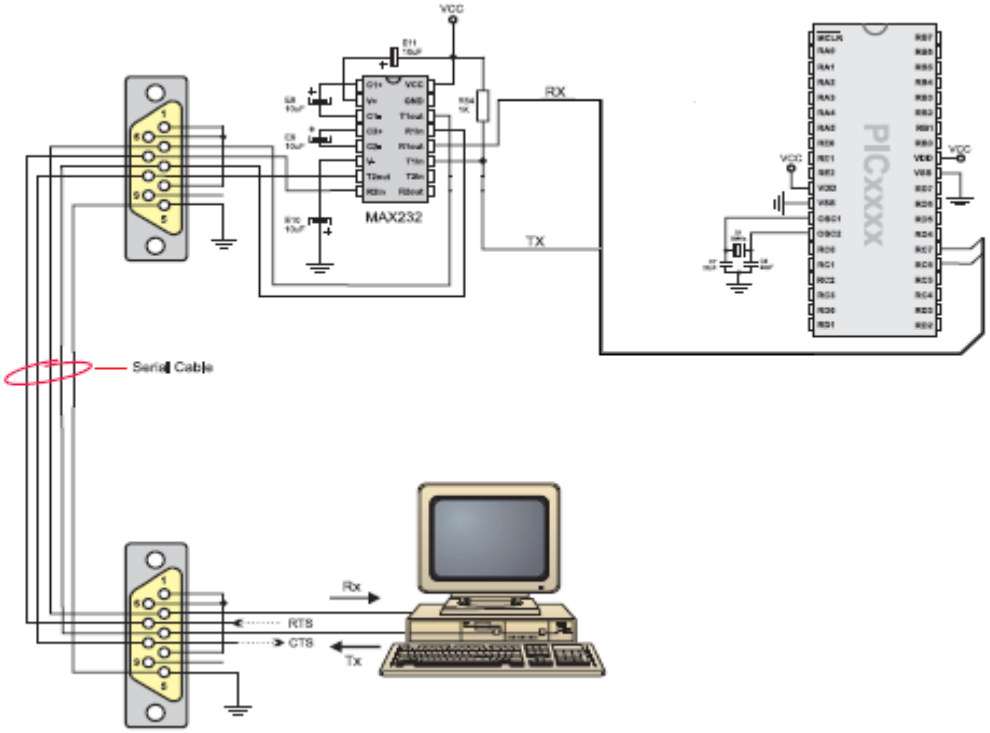
Fuente: elaboración propia.

Apéndice 14. **Circuito básico de conexión pantalla LCD y PIC**



Fuente: elaboración propia.

Apéndice 15. **Circuito básico de conexión puerto usart (serial) y pic**



Fuente: elaboración propia.

Apéndice 16. **Set de instrucciones asm**

Instrucciones orientadas al Byte									
Nemotecnico		Descripcion	Ciclos Necesarios						Bandera afecta
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff		C,DC,Z
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff		Z
CLRF	f	Clear f	1	00	0001	lfff	ffff		Z
CLRW	-	Clear W	1	00	0001	0xxx	xxxx		Z
COMF	f, d	Complement f	1	00	1001	dfff	ffff		Z
DECF	f, d	Decrement f	1	00	0011	dfff	ffff		Z
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		
INCF	f, d	Increment f	1	00	1010	dfff	ffff		Z
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff		Z
MOVF	f, d	Move f	1	00	1000	dfff	ffff		Z
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		C
NOP	-	No Operation	1	00	0000	0x0	0		C
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff		
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff		
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff		C,DC,Z
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff		Z
Instrucciones orientadas al Bit									
Nemotecnico		Descripcion	Ciclos Necesarios						Bandera afecta
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		
Instrucciones orientadas a las Constantes y Control									
Nemotecnico		Descripcion	Ciclos Necesarios						Bandera afecta
ADDLW	k	Add Literal and W	1	11	111x	kkkk	kkkk		C,DC,Z
ANDLW	k	AND Literal with W	1	11	1001	kkkk	kkkk		Z
CALL	-	Call Subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	k	Clear Watchdog Timer	1	0	0000	0110	0100		TO,PD
GOTO	k	Go to Address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR Literal with W	1	11	1000	kkkk	kkkk		Z
MOVLW	-	Move Literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	k	Return from Interrupt	2	0	0000	0000	1001		
RETLW	-	Return with Literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	0	0000	0000	1000		
SLEEP	k	Go into Standby mode	1	0	0000	110	11		TO,PD
SUBLW	k	Subtract W from Literal	1	11	110x	kkkk	kkkk		C,DC,Z
XORLW		Exclusive OR Literal with W	1	11	1010	kkkk	kkkk		Z

Fuente: elaboración propia.

Apéndice 17. **Set de instrucciones descripción**

ADDLW	**ADD Literal to W**
Sintaxis	ADDLW k
Palabras, Ciclos	1, 1
Operación	W + k -> W
Bit de estado	C, DC, Z
Descripción	Añade el contenido de W al contenido de k, y almacena el resultado en W

ADDWF	**ADD W to F**
Sintaxis	ADDWF f, d
Palabras, Ciclos	1, 1
Operación	W + f -> f si d=1 W + f -> W si d=0
Bit de estado	C, DC, Z
Descripción	Añade el contenido de W al contenido de f, y almacena el resultado en W si d=0, y en f si d=1

ANDLW	**AND Literal and W**
Sintaxis	ANDLW k
Palabras, Ciclos	1, 1
Operación	W AND k -> W
Bit de estado	Z
Descripción	Efectúa un AND lógico entre el contenido de W y el literal k, y lo almacena en W

ANDWF	**AND W with F**
Sintaxis	ANDWF f, d
Palabras, Ciclos	1, 1
Operación	W AND f -> f si d=1 W AND f -> W si d=0
Bit de estado	Z
Descripción	Efectúa un AND lógico entre el contenido de W y el contenido de f, y almacena el resultado en W si d=0, y en f si d=1

BCF	**Bit Clear F**
Sintaxis	BCF f, b
Palabras, Ciclos	1, 1
Operación	0 -> b(f)
Bit de estado	Ninguno
Descripción	Pone a cero el bit numero b de f

BSF	**Bit Set F**
Sintaxis	BSF f, b
Palabras, Ciclos	1, 1
Operación	1 -> b(f)
Bit de estado	Ninguno
Descripción	Pone a uno el bit numero b de f

BTFSC	**Bit Test, Skip If Clear**
Sintaxis	BTFSC f, b
Palabras, Ciclos	1, 1 o 2
Operación	Salta si b(f)=0
Bit de estado	Ninguno
Descripción	Si el bit numero b de f es nulo, la instrucción que sigue a esta se ignora y se trata como un NOP. En este caso, y solo en este caso, la instrucción BTFSC precisa dos ciclos para ejecutarse.

BTFSS	**Bit Test, Skip If Set**
Sintaxis	BTFSS f, b
Palabras, Ciclos	1, 1 o 2
Operación	Salta si b(f)=1
Bit de estado	Ninguno
Descripción	Si el bit numero b de f esta en uno, la instrucción que sigue a esta se ignora y se trata como un NOP. En este caso, y solo en este caso, la instrucción BTFSC precisa dos ciclos para ejecutarse.

CALL	**Subrutina Call**
Sintaxis	CALL k
Palabras, Ciclos	1, 2
Operación	En el caso de los 16c5X: PC + 1 -> Pila, k -> PC(0-7), 0 -> PC(8), PA2 a PA0 -> PC(9-11) En el caso de los 16c64, 71, 74, 84: PC + 1 -> Pila, k -> PC(0-10), PCLATH(3,4) -> PC(11,12)
Bit de estado	Ninguno
Descripción	Guarda la dirección de vuelta en la pila y despues llama a la subrutina situada en la dirección cargada en el PC. Atención: El modo de calculo de la dirección difiere según la familia PIC utilizada. También hay que posicionar bien PA2, PA1, PA0 (16c5X) o el registro PCLATH (en los demas PIC), antes de ejecutar la instrucción CALL

CLRF	**Clear F with F**
Sintaxis	CLRF f
Palabras, Ciclos	1, 1
Operación	00 -> f
Bit de estado	Z
Descripción	Pone el contenido de f a cero y activa el bit Z.

CLRW	**Clear W register**
Sintaxis	CLRW
Palabras, Ciclos	1, 1
Operación	00 -> W
Bit de estado	Z
Descripción	Pone el registro W a cero y activa el bit Z

CLRWDT	**Clear Watchdog Timer**
Sintaxis	CLRWDT
Palabras, Ciclos	1, 1
Operación	00 -> WDT y 0 -> predivisor del temporizador
Bit de estado	1 -> TO y 1 -> PD
Descripción	Pone a cero el registro contador del temporizador watchdog, asi como el predivisor

COMF	**Complement F**
Sintaxis	COMF f, d
Palabras, Ciclos	1, 1
Operación	-f -> f su d=1 -f -> W si d=0
Bit de estado	Z
Descripción	Hace un complemento de f bit a bit. El resultado lo almacena de nuevo en f si d=1 (borra el anterior), o en W si d=0 (f no varia)

DECF	**Decrement F to F**
Sintaxis	DECF f, d
Palabras, Ciclos	1, 1
Operación	f - 1 -> f si d=1 f - 1 -> W si d=0
Bit de estado	Z
Descripción	Decrementa el contenido de f en una unidad. El resultado se guarda en W si d=0 (f no varia), y en f si d=1

DECFSZ	**Decrement F, Skip If Zero**
Sintaxis	DECFSZ f, d
Palabras, Ciclos	1, 1(2)
Operación	f - 1 -> f si d=1 f - 1 -> W si d=0
Bit de estado	Ninguno
Descripción	Decrementa el contenido de f en una unidad. El resultado se guarda en W si d=0 (f no varia), y en f si d=1 Si el resultado es nulo, se ignora la siguiente instrucción y en este caso la instrucción dura dos ciclos.

GOTO	**Salto Incondicionalwith F**
Sintaxis	GOTO k
Palabras, Ciclos	1, 2
Operación	En el caso de los 16c5X: k -> PC(0-8), PA2 PA1, PA0 -> PC(9-11) En el caso de los 16c64,71,74 y 84: k -> PC(0-10), PCLATH(3,4) -> PC(11,12)
Bit de estado	Ninguno
Descripción	LLama a la subrutina situada en la dirección cargada en el PC

Fuente: elaboración propia.

ANEXOS

Anexo 1. **Hoja de especificaciones básicas**

MICROCHIP

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

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 other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

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
- 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 $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- 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

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

- Low-power, high-speed Flash/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf.

Consulta: 8 octubre 2017.

Anexo 2. **Hoja de especificaciones selección de pic**

PIC16F87XA PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	X	/XX	XXX
Device	Temperature Range	Package	Pattern

Device	PIC16F87XA(1), PIC16F87XAT(2), VDD range 4.0V to 5.5V PIC16LF87XA(1), PIC16LF87XAT(2), VDD range 2.0V to 5.5V	
Temperature Range	I	= -40°C to +85°C (Industrial)
Package	ML	= QFN (Metal Lead Frame)
	PT	= TQFP (Thin Quad Flatpack)
	SO	= SOIC
	SP	= Skinny Plastic DIP
	P	= PDIP
	L	= PLCC
	S	= SSOP

Examples:

a) PIC16F873A-I/P 301 = Industrial temp., PDIP package, normal VDD limits, QTP pattern #301.

b) PIC16LF876A-I/SO = Industrial temp., SOIC package, Extended VDD limits.

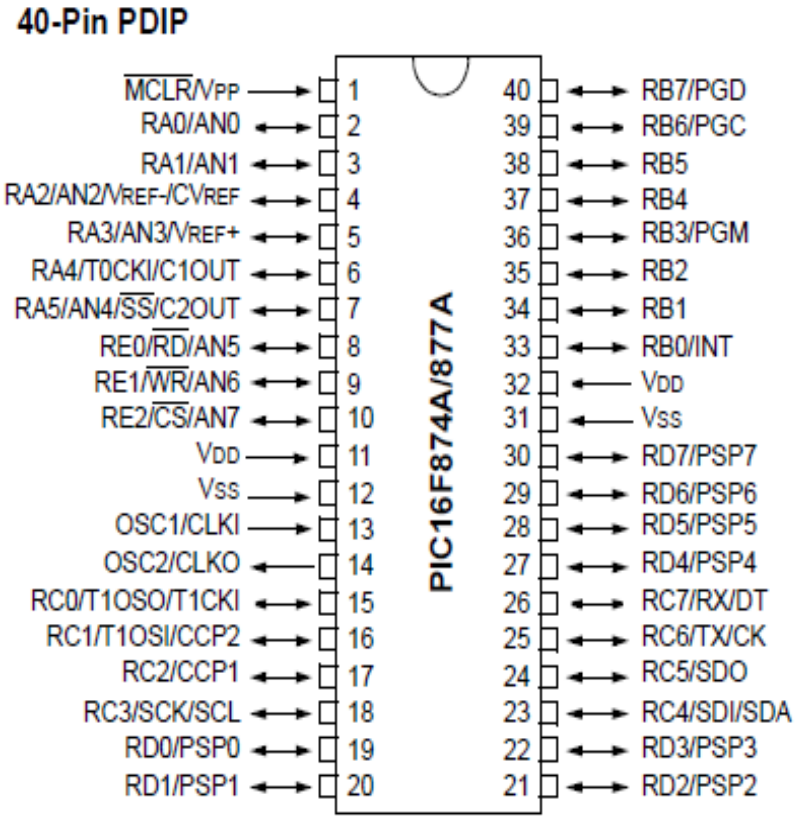
c) PIC16F877A-I/P = Industrial temp., PDIP package, 10 MHz, normal VDD limits.

Note 1: F = CMOS Flash
LF = Low-Power CMOS Flash
2: T = In tape and reel - SOIC, PLCC, TQFP packages only

Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf.

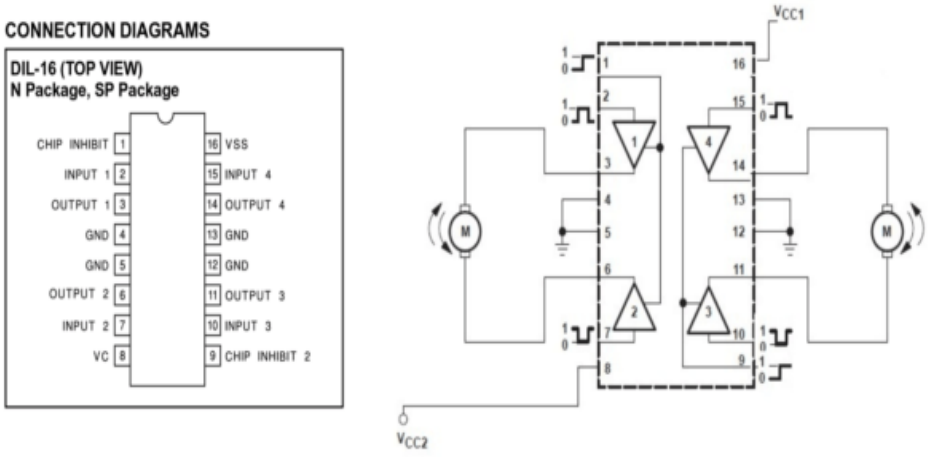
Consulta: 8 octubre 2017.

Anexo 3. **Configuración de pines físicamente**



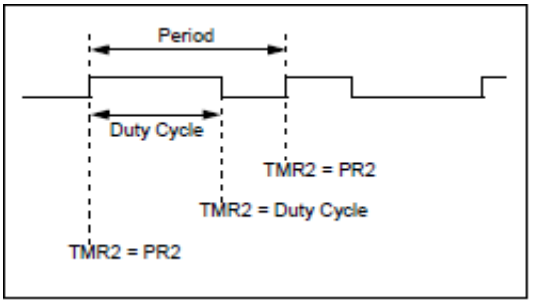
Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf. Consulta: 8 octubre 2017.

Anexo 4. **Circuito integrado de un puente h circuito básico giro motor dc**



Fuente: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf.
Consulta: 8 octubre 2017.

Anexo 5. **Valores para tren de pulsos PWM**



Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf.
Consulta: 8 octubre 2017.

Anexo 6. **Configuración de pantalla LCD**

Fuente https://www.geekfactory.mx/tutoriales/tutoriales-pic/pantalla-lcd-16x2-con-pic-libreria/.

Consulta: 20 septiembre 2017.

Anexo 7. **Configuración de pines pantalla LCD**

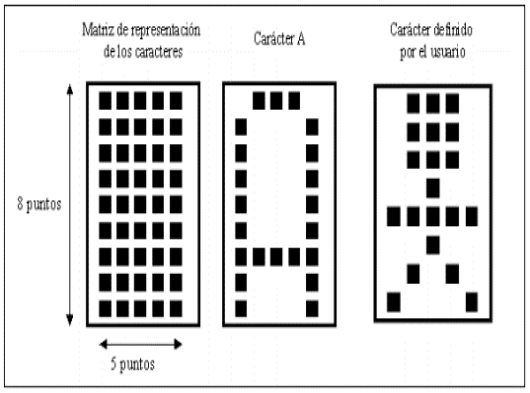
Instrucción	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear Display	0	0	0	0	0	0	0	0	0	1
Cursor Home	0	0	0	0	0	0	0	0		*
Entry Mode Set	0	0	0	0	0	0	0	1		S
Display On/Off control	0	0	0	0	0	0	1	D	C	B
Cursor /display shift	0	0	0	0	0	1	S/C	R/L	*	*
Funtion Set	0	0	0	0	1	DL	N	F	*	*
Set SGRAM	0	0	0	1	CGRAM address					
Set DDRAM	0	0	1	DDRAM address						
Read Busy-flag and address counter	0	1	BF	CGRAM/DDRAM address						
Write CGRAM or DDRAM	1	0	write data							
Read from CGRAM or DDRAM	1	1	read data							

I/D	1	Incremento
	0	decremento
S	1	desplaza el mensaje en la pantalla
	0	mesaje fijo en la pantalla
D	1	encender pantalla
	0	apagar pantalla
C	1	activar cursor
	0	desactivar cursor
B	1	parpadea carácter señalado por el curso
	0	no parpadea el carácter
S/C	1	desplaza pantalla
	0	mueve el cursor
RL	1	desplazamiento hacia la derecha
	0	desplazamiento hacia la izquierda
DL	1	bus de datos de 8 bits
	0	bus de datos de 4 bits
BF	1	indica operación interna del modulo
	0	indica fin de la operación interna del modulo

Fuente https://www.geekfactory.mx/tutoriales/tutoriales-pic/pantalla-lcd-16x2-con-pic-libreria/.

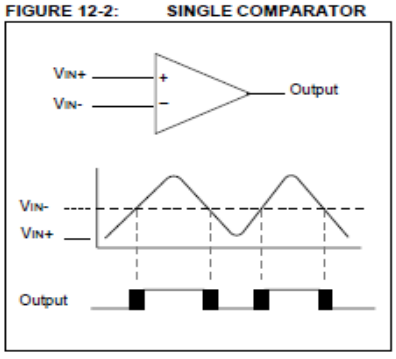
Consulta: 20 septiembre 2017.

Anexo 8. **Configuración matriz de visualización en una LCD**



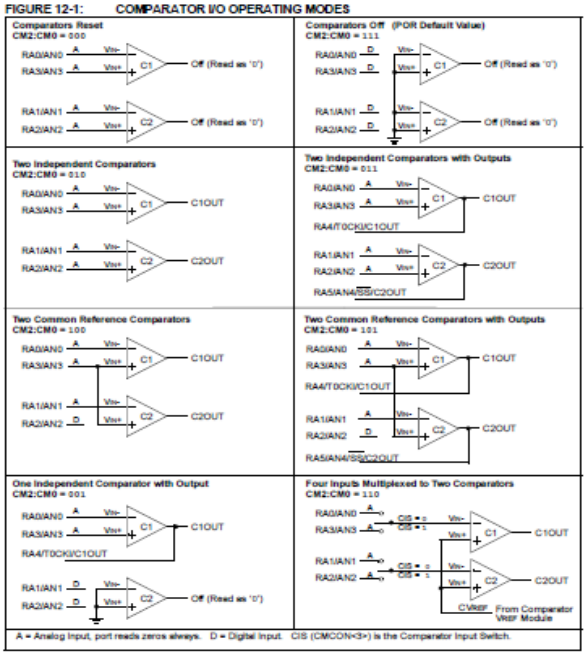
Fuente https://www.geekfactory.mx/tutoriales/tutoriales-pic/pantalla-lcd-16x2-con-pic-libreria/. Consulta 20 septiembre 2017.

Anexo 9. **Circuito basico comparador de voltaje AMP**



Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf. Consulta: 8 octubre 2017.

Anexo 10. **Configuraciones de AMP integrado en los PIC**



Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf.

Consulta: 8 octubre 2017.

Anexo 11. **Configuraciones de AMP integrado en los PIC**

REGISTER 12-1: CMCON REGISTER

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
bit 7							bit 0

bit 7 **C2OUT**: Comparator 2 Output bit
When C2INV = 0:
1 = C2 VIN+ > C2 VIN-
0 = C2 VIN+ < C2 VIN-
When C2INV = 1:
1 = C2 VIN+ < C2 VIN-
0 = C2 VIN+ > C2 VIN-

bit 6 **C1OUT**: Comparator 1 Output bit
When C1INV = 0:
1 = C1 VIN+ > C1 VIN-
0 = C1 VIN+ < C1 VIN-
When C1INV = 1:
1 = C1 VIN+ < C1 VIN-
0 = C1 VIN+ > C1 VIN-

bit 5 **C2INV**: Comparator 2 Output Inversion bit
1 = C2 output inverted
0 = C2 output not inverted

bit 4 **C1INV**: Comparator 1 Output Inversion bit
1 = C1 output inverted
0 = C1 output not inverted

bit 3 **CIS**: Comparator Input Switch bit
When CM2:CM0 = 110:
1 = C1 VIN- connects to RA3/AN3
C2 VIN- connects to RA2/AN2
0 = C1 VIN- connects to RA0/AN0
C2 VIN- connects to RA1/AN1

bit 2 **CM2:CM0**: Comparator Mode bits
Figure 12-1 shows the Comparator modes and CM2:CM0 bit settings.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf.
Consulta: 8 octubre 2017.

Anexo 12. **Configuraciones de ADC interno de un PIC**

FIGURE 13-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM

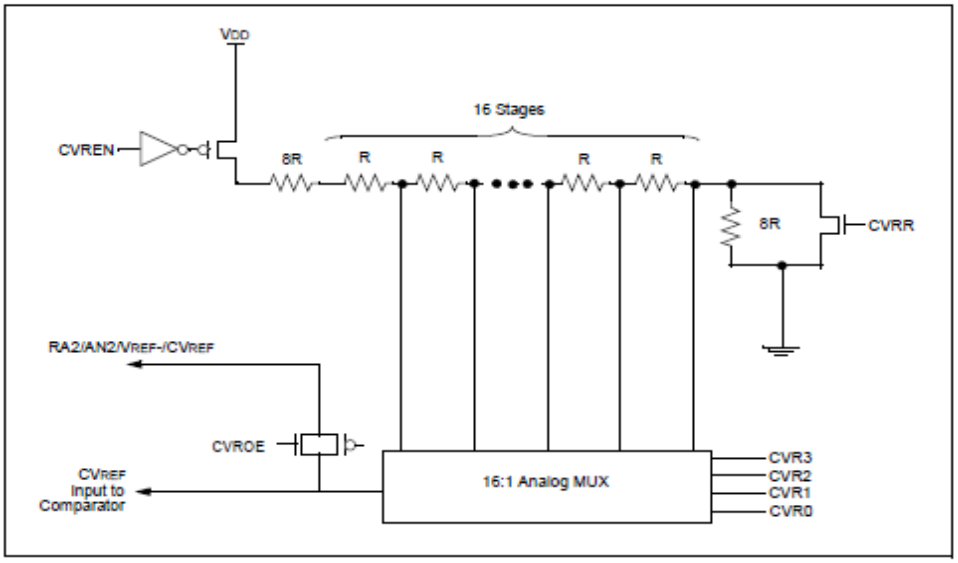


TABLE 13-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other Resets
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	000- 0000
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	0000 0111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'.
Shaded cells are not used with the comparator voltage reference.

Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf.

Consulta: 8 octubre 2017.