UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO

Ingeniería Mecánica Eléctrica

Control análogo y digital de servomotores con microcontroladores PIC serie 18FXXX

Tesis por el Alumno: Homero Jiménez Valderrama

Director de Tesis: Ing. José Manuel Ramírez Mora.
Asesores: Ing. Prócoro Pablo Luna Escorza.
Ing. Enrique García Guzmán.
Ing. Martín Meléndez Álvarez.
Ing. Abel Verde Cruz.

Ingeniería Mecánica Eléctrica área Eléctrica Electrónica

Agradecimientos

Está tesis está dedicada a la memoria de mi papá, a mi madre quien me infundió la ética y el rigor que guían mi transitar por la vida, A mis hermanos por confiar en mí. A mi Director de Tesis: Ing. Ramírez Mora José Manuel por su asesoramiento científico creyendo fuertemente en el desarrollo de proyectos de robótica en México y que me dio estímulo para seguir creciendo intelectualmente. Desgraciadamente mi papá no esta aquí para ver la consolidación de este trabajo, cosa que lamento infinitamente, y solo me consuela cumplir mi compromiso con él de finalizar este proyecto.

Quiero expresar un profundo agradecimiento Ing. Ramírez Mora José Manuel, director de ésta tesis, por toda su paciencia y atención, por toda la electrónica que me enseño y por su continuo apoyo durante estos años, tanto personal como académico. Ha sido un honor estar bajo su supervisión y aprender como debe ser un ser humano y un ingeniero.

Agradezco a mis sinodales la atención de revisar mi trabajo: Ing. Prócoro Pablo Luna Escorza. Ing. Enrique García Guzmán. Ing. Martín Meléndez Álvarez. Ing. Abel Verde Cruz. Por su apoyo y sus atenciones, y agradecerles por las continuas discusiones y su paciencia para contestar mis preguntas.

Quiero agradecer a los diferentes miembros de mi numerosa familia que siempre, de una u otra forma, nos han apoyado a mis padres, a mis hermanos y a mí.

Finalmente agradezco a la UNAM por la oportunidad que me ha dado de recibir una educación de alta calidad.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO

Pensamientos.

Escalera de la vida

Sube los escalones de tu existencia

Despacio, cauteloso, con mucha calma, inteligencia
Y buena voluntad, sube los escalones

Pensando siempre en la gloria que se encuentra
En lo mas alto de la escalera que estas subiendo

No cedas ni un segundo al desánimo
No permitas que la indecisión te domine.

Aprende a superarlos. El mundo pertenece a
Los seres optimistas, positivos y sinceros;
Nunca será de los cobardes, quejosos,
Indecisos, mentirosos y deshonestos,

Estos últimos se quedan en los primeros escalones
De la gran escalera, Prosigue en línea recta,
Buscando tus sagrados objetivos.

Acuérdate. la victoria es de los que luchan
Contra las situaciones desfavorables,
Sin perder el vigor, la fe, y el ideal de la vida

Si no vences es porque te dejaste contaminar
La ola negra del mal y perdiste el deseo
De luchar hasta el final porque, quien lucha,
Dando el verdadero esplendor a la vida
Al bien y persiste sin retroceder... ¡Vencerá!

INDICE

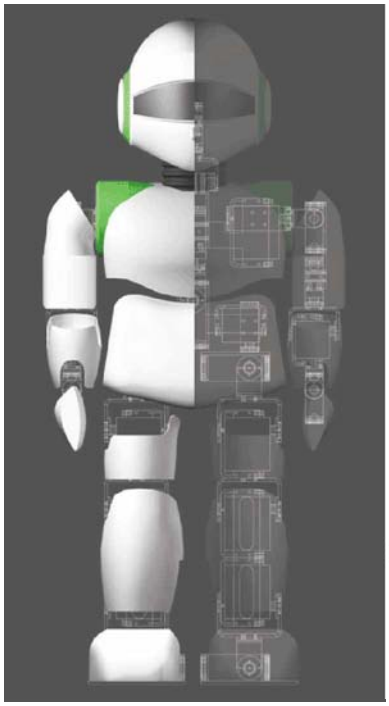
Introducción

Con esta tesis quiero resaltar la importancia de los servomotores en la robótica con ejemplos didácticos y reales, para esto utilizo elementos fáciles de conseguir y de realizar. Empezamos con elementos análogos como el Timer NE555 hasta llegar con el microcontrolador PIC16F877, CPLD, Display 16x2, Display grafico 64x128 con PIC18F452. Uso del lenguaje ensamblador, y Let Pic Basic.

El servomotor es un dispositivo electromecánico que convierte de una señal análoga en ancho de pulso a una posición en grados que van de 0 a 180º con una retroalimentación constante de 50 Hz que rectifica la posición. Es de suma importancia este dispositivo en la robótica porque nos asegura una posición en ángulo a la cual queremos que se coloque nuestro mecanismo.

Acerca de la portada este es un humanoide hecho con un Dispositivo Lógico Programable (PLDs). Cuenta con 25 servomotores que le dan movimiento a todo el robot que hace que parezca que tiene vida.

Figura 1.1. En esta imagen se puede apreciar el robot llamado pino hecho en Japón año 2004 en el cual se muestra la charcaza y el mecanismo que le permite moverse.



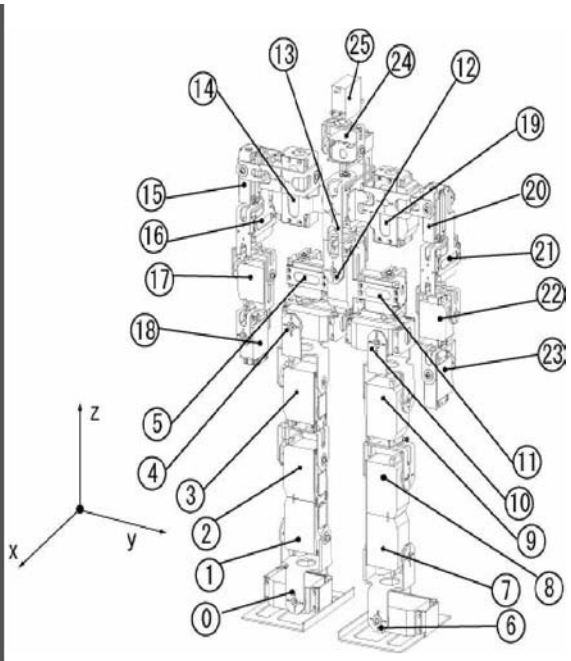


Figura 1.2 en esta figura se pueden apreciar los 25 servomotores que le dan vida a este robot.

Prefacio

En esta obra quiero introducir y dejar una ayuda para mis futuros compañeros en la carrera de IME, que vean que es real y fácil controlar un servomotor que es muy útil en la programación de robótica, esta obra va dirigida a los estudiantes de los últimos semestres de ingeniería, subraya los principios y aplicaciones fundamentales de los servomotores, esta obra la eh revisado varias veces, quiero aclarar que cualquier persona que no haya estudiado un nivel de licenciatura puede seguir los pasos para hacer los programas si es una persona del tipo autodidacta puesto que capture los programas de la pantalla compilados y sin error de sintaxis. También dejo unos videos en el CD adjunto donde se ve la muestra de los robots para que vean que es posible hacerlo realidad. En esta obra explico un tanto como entender al microcontrolador en este caso PIC pero de alguna forma parecida se pueden programar los diferentes microcontroladores en el mercado.

Figura 1.3. Robot pino ya terminado. En esta figura se aprecia a pino el humanoide ya terminado.

Prologó

En esta tesis quiero resaltar la facilidad del control de los servos y sus múltiples utilidades, también la facilidad de la programación de un microcontrolador. Acerca de de la carátula de esta tesis quiero mencionar que es un humanoide echo en Japón en el 2004, esta tesis tiene fecha del año 2006, pero al ver el noticiario que sorprendió a muchos ese robot echo con servomotores quiero hacer referencia que no es difícil controlarlos y no solo dejo un uso si no varias formas de modificar el uso. En estos experimentos trato de usar todo los materiales mas comunes para que el lector pueda hacerlos y repetirlos.

Resumen

En el primer capitulo describo la señal necesaria para que el servomotor funcione. No es mas que una señal cuadrada que varia su ancho de pulso y se refresca a la velocidad de 50 Hz.a 100 Hz. Y solo usa un pequeño ancho de pulso de 0.5 milisegundos a 2.5 milisegundos de los 20 milisegundos disponibles. ¿Para que dejar tanto tiempo sin uso? Ese tiempo es para que con solo un microcontrolador pueda controlar hasta 8 servos en dado caso que el micro no tenga interrupciones pero el microcontrolador que usamos en esta obra va desde el que no tiene interrupción de ancho de pulso asta el que si tiene hasta dos interrupciones de ancho de pulso (PWM – Pulse Wave Modulation).

En el segundo capitulo describo las fichas técnicas para los microcontroladores en el mercado, y un circuito análogo a digital y un traductor que nos mostrara el resultado en sistema decimal con un sencillo programa. Este nos es útil para las personas que no podemos traducir de binario a decimal a simple vista que creo que somos la mayoría. Y un segundo circuito que nos describe como controlar análogamente un servomotor por un sencillo timer 555. que son de los primeros circuitos que nos enseñan en la carrera. Hay una referencia sobre los distintos programadores que usamos en esta tesis.

En el tercer capitulo describo las principales funciones de los displays asi como el cristal líquido es un tipo especial de estado de agregación de la materia que tiene propiedades de las fases líquidas y la sólida. Dependiendo del tipo de cristal líquido, es posible, por ejemplo, que las moléculas tengan libertad de movimiento en un plano, pero no entre planos, o que tengan libertad de rotación, pero no de traslación. Si estas moléculas tienen un marcado momento bipolar eléctrico, su orientación puede ser controlada mediante campos eléctricos. Algunas de estas moléculas nemáticas presentan propiedades ópticas según su orientación permitiendo o impidiendo el paso de la luz o actuando sobre su polarización. Su aplicación más directa es para la fabricación de pantallas de cristal líquido.

En el cuarto capitulo definimos y comentar la parte de la recepción de la señal infrarroja ya que, por un lado, es sumamente sencilla de conectar a un microcontrolador y, por otro, es la que nos va a obligar a diseñar y ajustar los circuitos que necesitamos en la parte de la emisión. Para la recepción vamos a utilizar un dispositivo que unifica en el mismo encapsulado el receptor de luz infrarroja, una lente y toda la lógica necesaria para distinguir señales moduladas a una determinada frecuencia. Concretamente, en este montaje utilizaremos los receptores IS1U60 de Sharp que se activan cuando reciben una luz infrarroja modulada a una frecuencia de 38 kHz (el haz infrarrojo se apaga y enciende 38000 veces por segundo). Esto los hace compatibles con un gran número de mandos a distancia de electrodomésticos.

En el quinto capitulo describimos las principales características de los microcontroladores mas comunes como para el PIC16F84a las características son dos puertos A y B, el puerto A esta compuesto por 5 pines del pin 0 al 4, el pin llamado RA4/T0CKI es utilizado para un reloj externo o una salida, cuando configuramos este pin como salida tenemos que colocarle una resistencia a tierra, el puerto B tiene la característica que se puede configurar como entrada y salidas, pero cuando están configuradas como entrada se le puede añadir la característica de resistencia de pull – up, esto hace que se dispare a 5 voltios cuando no es 0. Este PIC es parecido al 16F84 pero con la diferencia principal entre muchas que tiene por hardware una interrupción llamada HPWM, modulación de ancho de pulsos. PIC16F876 Este microcontrolador aparte de tener HPWM y las características de los otros PICs, tiene 6 convertidores análogos a digital en el puerto A. PIC18F452 Este PIC tiene 8 convertidores análogo digital en el puerto A y E, 5 en el puerto A y 3 en el puerto E aparte de todas las características anteriores tiene doble HPWM, entre otras características.

En el sexto capitulo se describe como empezar a programar el pic. Sí tu escribes un punto y coma (;) en cualquier lugar en tu programa, el compilador ignorará todo lo que este después hasta el retorno de carro. esto significa que nosotros podemos adherir comentarios en nuestro programa para recordarnos de que nosotros estábamos haciendo en el lugar primero. Es bueno practicar, desde los más simples programas. Tu podrías entender totalmente bien como yo programa trabaja ahora, pero en pocos meses de tiempo, tu podrías estar rascándote la cabeza, entones usa los comentarios donde tu puedas aquellos no tienen limites. Segundo. tu puedes asignar nombres a las constantes vía registros (mas acerca de eso después) tómalo hasta la distancia que sea fácil leerlo en ingles que estas escribiendo, o que valor es. Tercera, pon algo de líneas de cabecera en tus programas usando las líneas intermedias. Autor Fecha Versión Titulo Descripción.

En el séptimo capitulo hablo sobre unos ejemplos didácticos sobre mini robótica término robótica procede de la palabra robot. La robótica es, por lo tanto, la ciencia o rama de la ciencia que se ocupa del estudio, desarrollo y aplicaciones de los robots. Otra definición de robótica es el diseño, fabricación y utilización de máquinas automáticas programables con el fin de realizar tareas repetitivas como el ensamble de automóviles, aparatos, etc. y otras actividades. Básicamente, la robótica se ocupa de todo lo concerniente a los robots, lo cual incluye el control de motores, mecanismos automáticos neumáticos, censores, sistemas de cómputos, etc.

El término robótica se le atribuye a Isaac Asimov. Los tres principios o leyes de la robótica según Asimov son:

Un robot no puede lastimar ni permitir que sea lastimado ningún ser humano.

El robot debe obedecer a todas las órdenes de los humanos, excepto las que contraigan la primera ley.

El robot debe autoprotegerse, salvo que para hacerlo entre en conflicto con la primera o segunda ley.

Generalidades de los servomotores I

Servomotores

¿Que es un servomotor?

Un servomotor es un dispositivo electromecánico en el cual una retroalimentación determina la posición del rotor. Sí la tensión vence la posición del rotor el servomotor tiene como característica que en cuanto se liberé la tensión regresar a su posición original. Los servomotores son usados frecuentemente dentro de los robots y carros a radio control, planeadores y botes.

¿Como funcionan los servomotores?

Para hacerlos funcionar se necesita una señal variable en ancho de pulso que va de los 50hz a 100hz máximo. Para hacerlo posicionar en la posición 0 grados de giro debemos mandar una señal aproximada a los 0.5 mili segundos en estado alto y en estado bajo 19.5 mili segundos la suma de 20 milisegundos o sea 50hz.

En la primera grafica se ven los pulsos enviados a los servomotores la frecuencia es de 52hz, valor pico de 4.63v. De acuerdo con la hoja de datos deben ser 4.5V a 6V máximo y frecuencia de 50hz a 100hz máximo.

En el pulso superior de la figura 1.4 es un estado normal de 1.5 milisegundos en estado alto mientas en el pulso inferior de la figura 1.4 es un tiempo de 0.5 milisegundos en estado alto, el estado bajo es muy tolerable. Esto hace que el servo se coloque en la posición 0 grados.

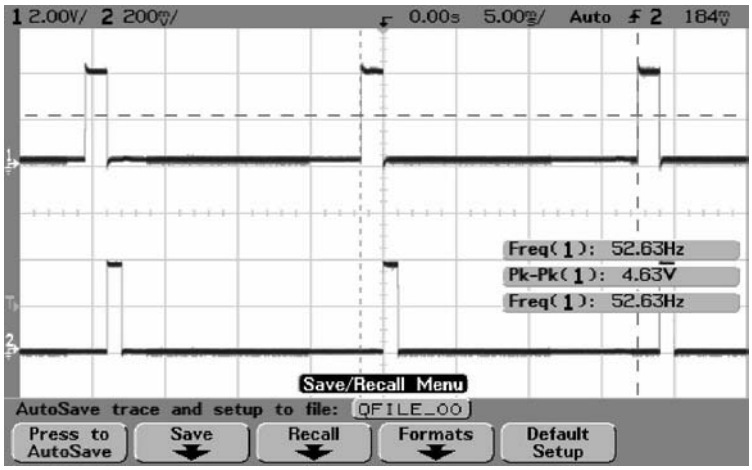


Fig. 1.4. En esta imagen se hace una comparación del ancho de pulso de un estado normal de 1.5 milisegundos a un estado 0.5 milisegundos. (Imagen tomada del osciloscopio Agilent del laboratorio de IME)

En el pulso superior es un estado normal de 1.5 milisegundos en estado alto lo sigo dejando así para que se vea la comparación mientas en la grafica de abajo es un tiempo de 1.5 milisegundos en estado alto, el estado bajo es muy tolerable. Esto hace que el servo se coloque en la posición 90 grados.

Nota: aunque el servomotor reciba un estado bajo y alto no quiere decir que es digital, realmente es un dispositivo análogo puesto que se controla con el ancho de pulso. Con un microcontrolador podemos emitir un ancho de pulso controlado digitalmente pero esto no hace un servomotor digital.

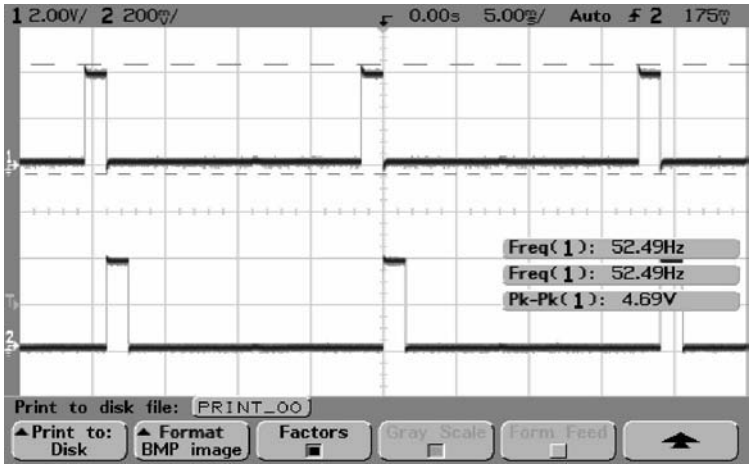


Fig. 1.5. En esta imagen se hace una comparación del ancho de pulso de un estado normal de 1.5 milisegundos a un estado 0.5 milisegundos. (Imagen tomada del osciloscopio Agilent del laboratorio de IME)

En el pulso superior es un estado normal de 1.5 milisegundos en estado alto lo sigo dejando así para que se vea la comparación mientas en la grafica de abajo es un tiempo de 2.5 milisegundos en estado alto, el estado bajo es muy tolerable. Esto hace que el servo se coloque en la posición 180 grados.

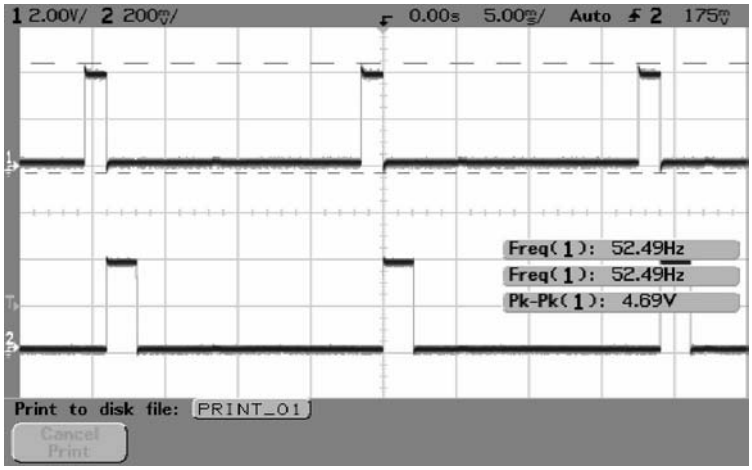


Fig. 1.6. En esta imagen se hace una comparación del ancho de pulso de un estado normal de 1.5 milisegundos a un estado 0.5 milisegundos. (Imagen tomada del osciloscopio Agilent del laboratorio de IME)

Modificar un servomotor para que gire infinitamente

	Tren de Pulsos	Motor
A	pulso bajo	Gira motor hacia la izquierda
B	Pulso medio	Gira el motor hacia el centro
C	pulso alto	Gira el motor hacia la derecha

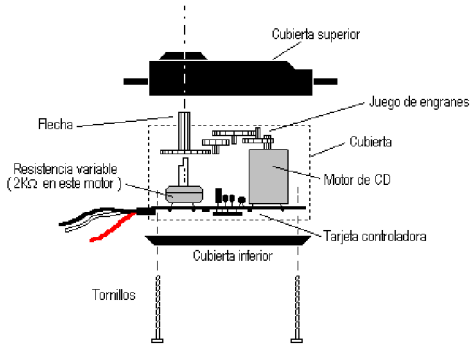


Fig. 1.7 estructura de un servomotor.

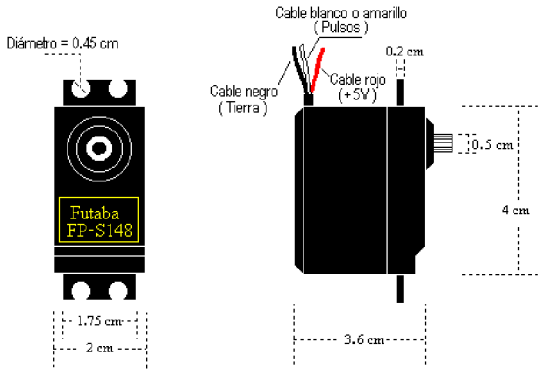


Fig. 1.8. Terminales de un servomotor

Fig. 1.9 Foto de un servomotor modelo S3003

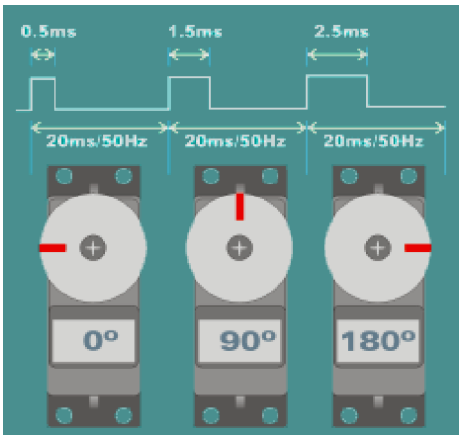


Fig. 1.10 Datos técnicos del funcionamiento de un servomotor

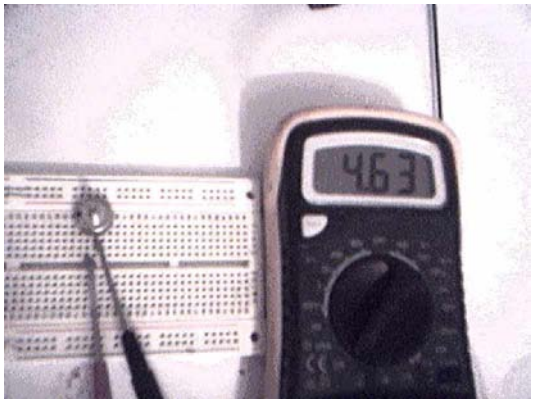
Primero tenemos que quitar el tope físico para que nuestro servo gire infinitamente, con un cutre será más que suficiente para cortar este tope. Recomiendo el servomotor 3004 aunque en la fig. 1.11 esta el 3003. Las diferencias entre estos dos son que el 3004 tiene un balero que ayuda a soportar peso que el 3003 no tiene.

Trata de usar guantes de medico ya que se nos advierte que los aceites usados en ese mecanismo son cancerigenos y en mujeres pueden tener defectos de nacimiento en sus hijos

Fig. 1.11. Se muestra el tope físico de un servomotor

Después tenemos que quitar el potenciómetro que también limita el giro, este lo tenemos que sustituir por dos resistencias equivalentes, podemos solamente sumirlo para que no estorbe, pero para controlar mejor el servo tenemos que sustituirlo por 2 resistencias de precisión o metálicas para que no haya variaciones con respecto al otro servomotor del lado derecho que pondremos en nuestro robot.

Fig. 1.12. Se muestra el tope físico resistivo de un servomotor

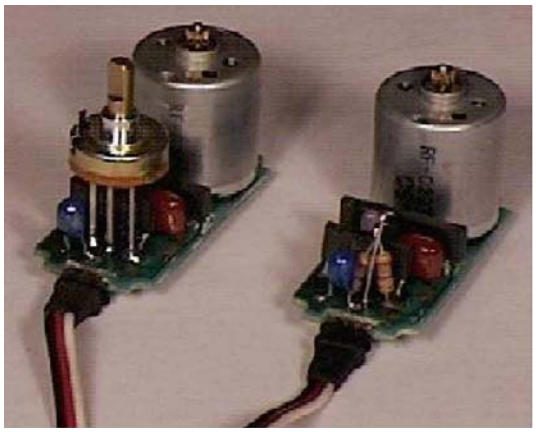


Tenemos que le potenciómetro extraído tiene un valor aproximado como máximo 4.63 kilos ohm.

Ósea que la división nos da 2.3 kilo ohm y la mas próxima en menor es 2.2 y en mayor 2.7

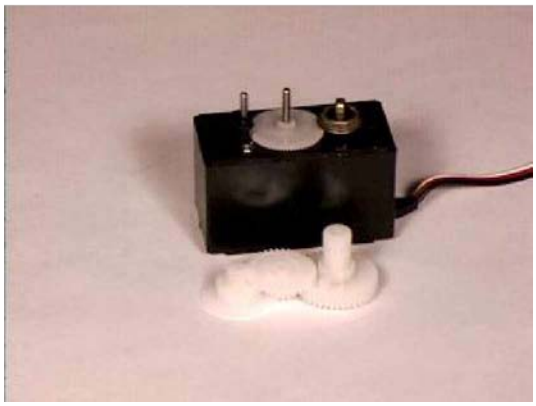
1.13. Se muestra el valor en ohms de la resistencia giratoria un servomotor

Ahora cambiamos el potenciómetro por dos resistencias del valor de la mitad del potenciómetro como se muestra en la figura. Tenemos que en resistencias de película metalizada o metal film al 1% de tolerancia a medio watt comercialmente solo existen en menor valor en 2.2 kilo ohm y en mayor 4.7 kilo ohm. Así que utilizamos 2.2 kilo ohm.



Debe quedarte algo así como en la foto.

1.14. Se hace referencia a como debe quedar las dos resistencias que sustituyen a la resistencia variable de un servomotor.



Ahora tenemos que volverlo a armar y cerrar, para que te acuerdes como se arma te pongo otra vez la foto. Repite los pasos para el segundo motor.

1.15. Se muestra el orden del engranaje del servomotor

Circuitos básicos II

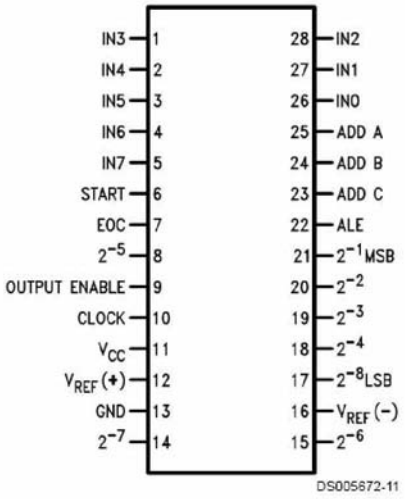
Fichas técnicas de los circuitos.

ADC0808

Este chip nos va a servir para convertir de análogo a digital pero como en digital es algo lento traducirlo a decimal con la ayuda de los decodificadores Cypress y 74LS47 es posible ver el resultado en decimal.

Connection Diagrams

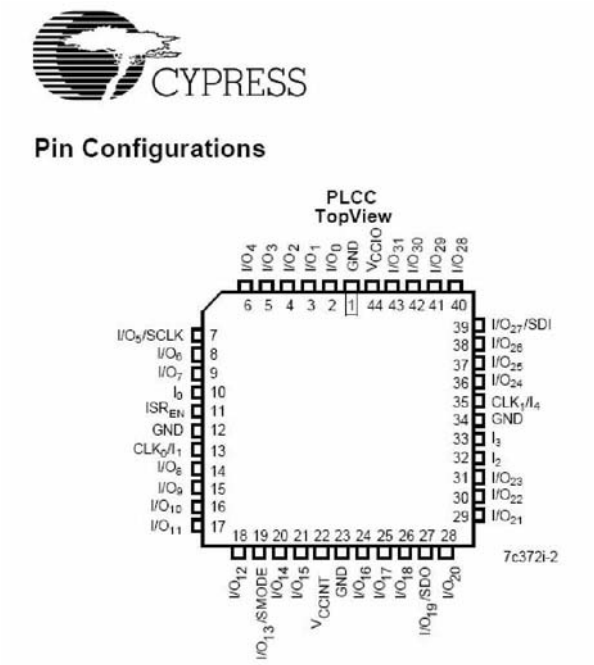
Dual-In-Line Package



Order Number ADC0808CCN or ADC0809CCN
See NS Package J28A or N28A

Fig. 2.1. Se muestra las terminales del circuito ADC0808

CY7C372i



Este circuito nos sirve para usar la lógica residual que necesitemos en este experimento para cada Display separamos 4 bits de un total de 3 entonces hacemos un programa que nos decodifique esto. El programa esta a continuación.

Fig. 2.2. Se muestra las terminales del circuito Cy7C372i

Tenemos que hacer una conversión de binario a decimal para esto hacemos un decodificador de binario a decimal. En el caso de 10100101 que en decimal es 165 tendremos que separar en binario el 1 el 6 y el 5 en binario que seria 01, 0110,0101 así que ponemos un código que diga cuando el numero sea 10100101 entonces el vector d será 0101100101 y así con todos los números hasta llegar al 254 y al final colocamos en caso de otros valores el numero 255 que en binario es 11111111 será 01,0101,0101.

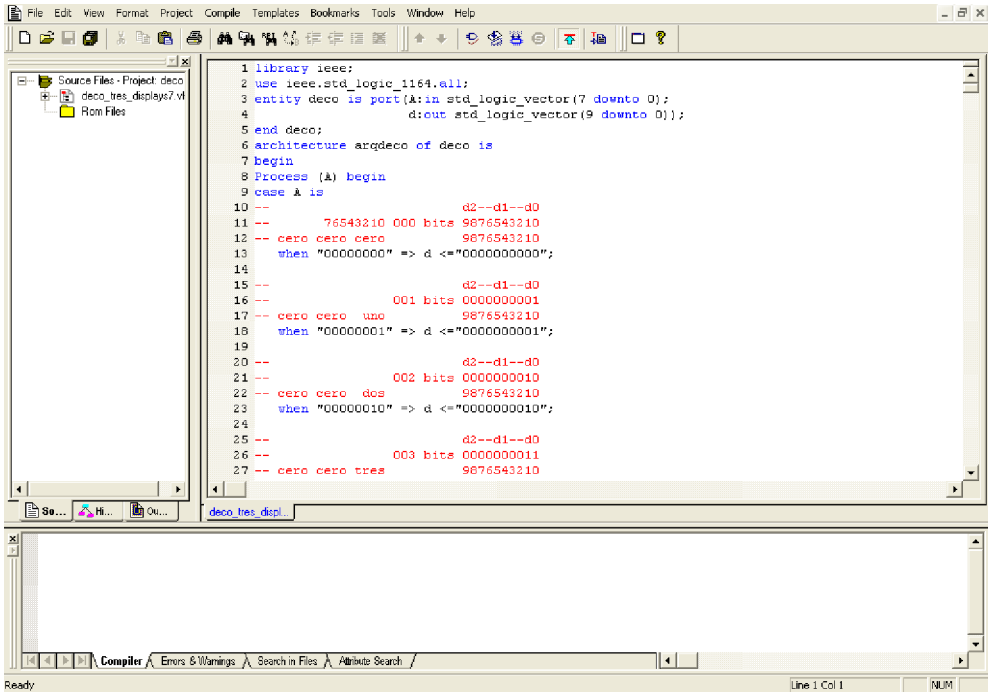


Fig. 2.3. Se muestra parte del programa para hacer la conversión binaria a decimal para 8 bits

Así consecutivamente vamos cambiando el código correspondiente hasta llegar a 254 y el ultimo que es el caso de que todos sean unos nos corresponde el numero 1001010101.

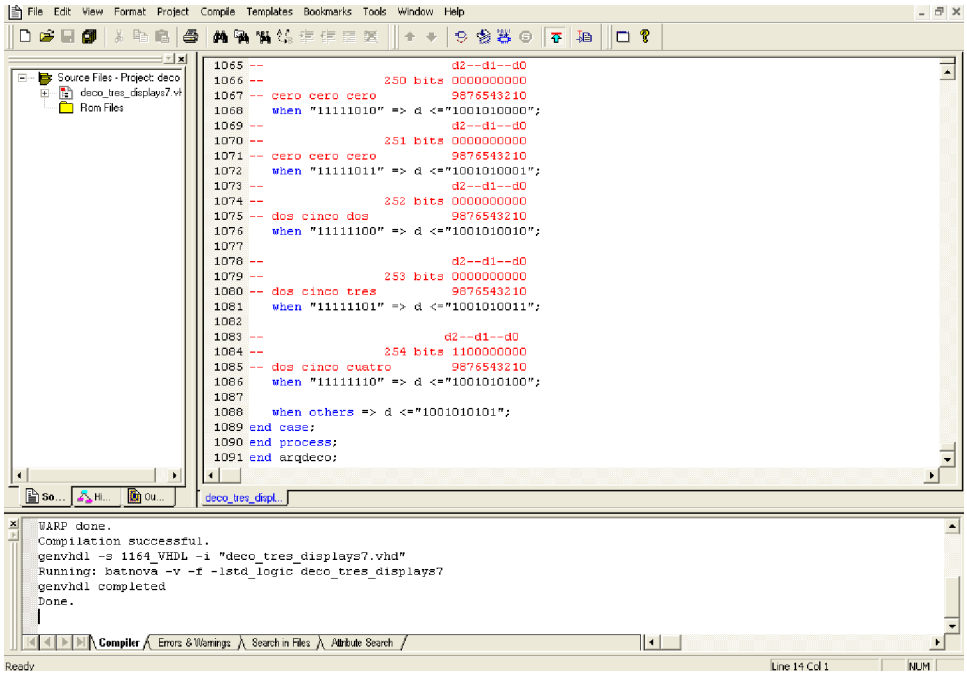


Fig. 2.4. Se muestra la parte final del programa para hacer la conversión binaria a decimal para 8 bits

74LS47

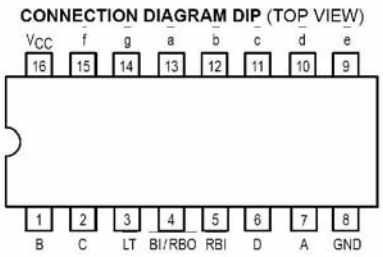


Fig. 2.5. Se muestran las terminales del circuito 74LS47 necesario para convertir los 4 bits a un display decimal.

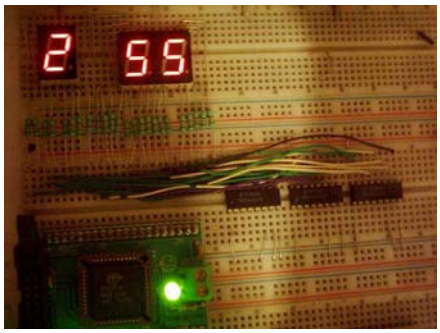


Fig. 2.6. Se muestra ya funcionando el programa y le chip y los displays de 7 segmentos en la tableta de prototipo.

Este es un circuito que nos decodifica de un numero binario a decimal. Para saber que numero en decimal estamos en el convertidor analógico a digital pero como solo puede trasformar de 4 bits de 0 a 9 en un Display de 7 segmentos usamos 3 y con el circuito siguiente convertimos de 8 bits a 3 segmentos de 4 bits.

Control análogo de los Servomotores

Ahora haremos los cálculos necesarios para hacer los temporizadores del CI 556, este es un temporizador doble, tendremos que generar un tiempo de estado alto y uno bajo, el único inconveniente es que ha este CI solo se le puede aumentar el estado alto pero no el bajo a si que lo meteremos a un inversor para obtener los tiempos deseados.

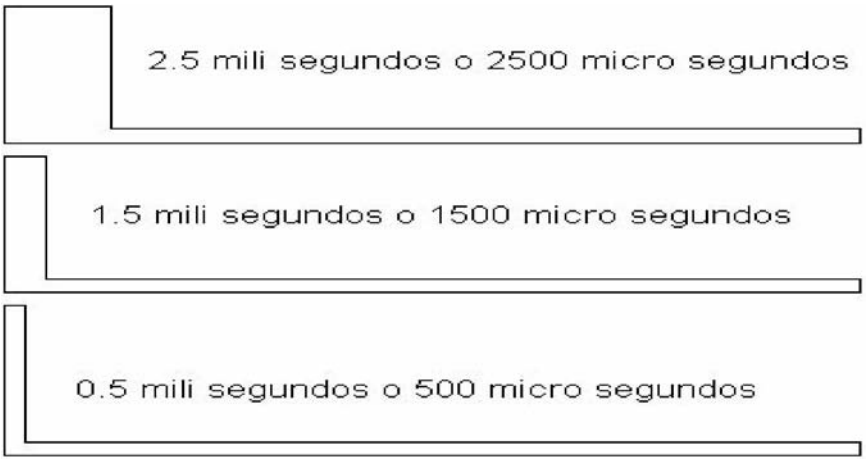


Fig. 2.7. Se muestra en proporción el tamaño del ancho de pulso para controlar un servomotor

La función especial del temporizador es que al cargarse del 30 al 60 % del voltaje total introducido es un periodo alto y mientras se descarga del 60% al 30% es un estado bajo.

Esto lo usamos para dale rango de tolerancia a nuestro infrarrojo puesto que si no fuera así al mínimo cambio de luz haría un salto repentino en la posición.

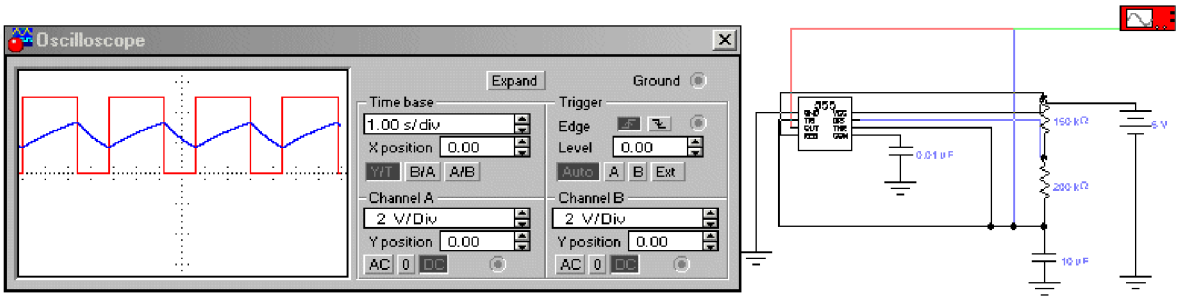


Fig. 2.8. Se muestra una simulación con Electronics Work Bench.

Programadores

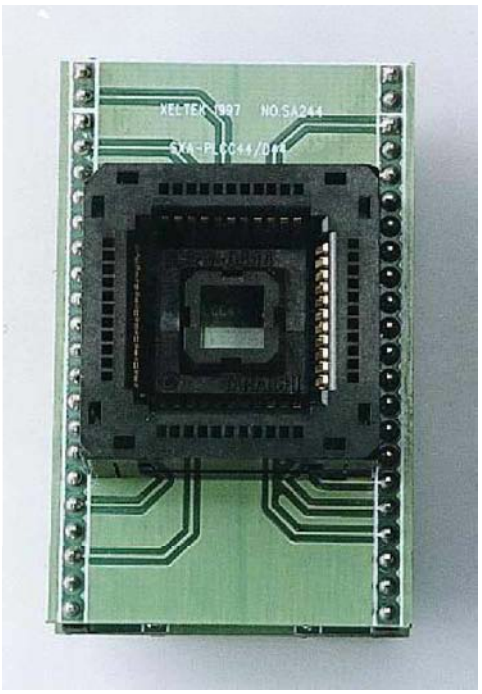
Aparte del programador PIC START Plus! Hay otros programadores cuyo software es aparte del ofrecido en la página de Microchip. Si el encapsulado es PLCC 44, tenemos que recurrir a un adaptador para programarlo en la figura 2.11.

Fig. 2.9. Se muestra el programador universal

Fig. 2.10. Se muestra el programador Pic Start Plus!

Fig. 2.11. Se muestra el programador Cypress 44

Fig. 2.11. Se muestra el adaptador para chips PLCC



Displays de cristal liquido III

Cristal liquido

Se suele atribuir el descubrimiento de los cristales líquidos al botánico F. Reitinzer que en 1888 encontró una sustancia que parecía tener dos puntos de fusión. Un año más tarde Otto Lehmann solventó el problema con la descripción de un nuevo estado de la materia intermedio entre un líquido y un cristal.

La principal característica de estos compuestos es que sus moléculas son altamente anisotrópicas en su forma, pueden ser alargadas, en forma de disco u otras más complejas como forma de banana. En función de esta forma el sistema puede pasar por una o más fases intermedias (mesofases) desde el estado cristalino hasta el líquido. En estas mesofases el sistema presenta propiedades intermedias entre un cristal y un líquido. Dos de las principales fases de un cristal líquido son la fase nemática y la esméctica. En la fase nemática los centros de masas de las moléculas están colocados como en un líquido (sin orden de largo alcance) y al menos uno de los ejes principales de las moléculas apunta, en promedio, a lo largo de una determinada dirección (llamada director). En la fase esméctica, al igual que en la nemática, tenemos orden de largo alcance orientacional y además los centros de masas moleculares están organizados en capas a lo largo de una dimensión. El eméctico, por tanto, presenta también orden de largo alcance posicional en una dimensión.

El cristal líquido es un tipo especial de estado de agregación de la materia que tiene propiedades de las fases líquidas y la sólida. Dependiendo del tipo de cristal líquido, es posible, por ejemplo, que las moléculas tengan libertad de movimiento en un plano, pero no entre planos, o que tengan libertad de rotación, pero no de traslación.

Los llamados cristales líquidos termo trópicos están compuestos generalmente por moléculas con formas de cilindros o discos. Según la temperatura y tipo de moléculas, los cristales líquidos termo trópicos pueden organizarse en diferentes fases: nemáticas, colestéricas, esmécticas o columnares. Si estas moléculas tienen un marcado momento bipolar eléctrico, su orientación puede ser controlada mediante campos eléctricos.

Algunas de estas moléculas nemáticas presentan propiedades ópticas según su orientación permitiendo o impidiendo el paso de la luz o actuando sobre su polarización. Su aplicación más directa es para la fabricación de pantallas de cristal líquido.

Otra categoría existente es la de los cristales líquidos liotrópicos, formados por agregados de moléculas anfifílicas (moléculas que poseen en su misma estructura, regiones hidrofóbicas e hifrofílicas) cuando colocadas en un medio polar (agua) o apolar (solvente orgánico).

Display 16x2

Fig. 3.1. Se muestra un display de 16 caracteres de largo por dos líneas de alto.

Explicación detallada de los pines.

Pin 1 V_{DD} este pin se conecta al Voltaje 0
Pin 2 Vss este pin se conecta a Voltaje positivo a 5 Volts
Pin 3 Vo este pin se conecta a un voltaje variable que va de 0 a 5 Volts para variar el contraste.
Pin 4 RS (D/I) este pin indica que si esta en alto los datos en DB0 a DB7 son datos y si el pin esta en bajo el contenido en la entrada de los pines son instrucciones.
Pin 5 R/W este pin indica que si esta en alto en los pines de DB0 a DB7 podemos leer o escribir en ellos.
Pin 6 E Pin de enable que nos dice que en el momento de un salto de alto a bajo son leídos todos pines de entrada o salida.
Pin 7 DB1 Pin no 0 para entrada de datos
Pin 8 DB1 Pin no 1 para entrada de datos
Pin 9 DB2 Pin no 2 para entrada de datos
Pin 10 DB3 Pin no 3 para entrada de datos
Pin 11 DB4 Pin no 4 para entrada de datos
Pin 12 DB5 Pin no 5 para entrada de datos
Pin 13 DB6 Pin no 6 para entrada de datos
Pin 14 DB7 Pin no 7 para entrada de datos
Pin 15 A Pin para alimentar la luz de fondo positivo a 5 Volts
Pin 16 K Pin para alimentar la luz de fondo negativo a 0 Volts

Características.

Este display es capaz de mostrar cada uno de los caracteres mostrado en la siguiente tabla.
Con simplemente poner en el cuerpo del programa la instrucción PRINT y entre comillas lo que queremos que se escriba en el display y si queremos colocar el cursor en cierta posición del display solo basta con colocar después de la instrucción PRINT la posición en numero de la casilla disponible separado por una coma en X y Y siendo para la primera fila Y y para las columnas X superior a inferior y de 16 caracteres por 2 filas. Ejemplo:

PRINT 1,1 “HOLA MUNDO”
PRINT 2,1 “SOY YO”

Tabla

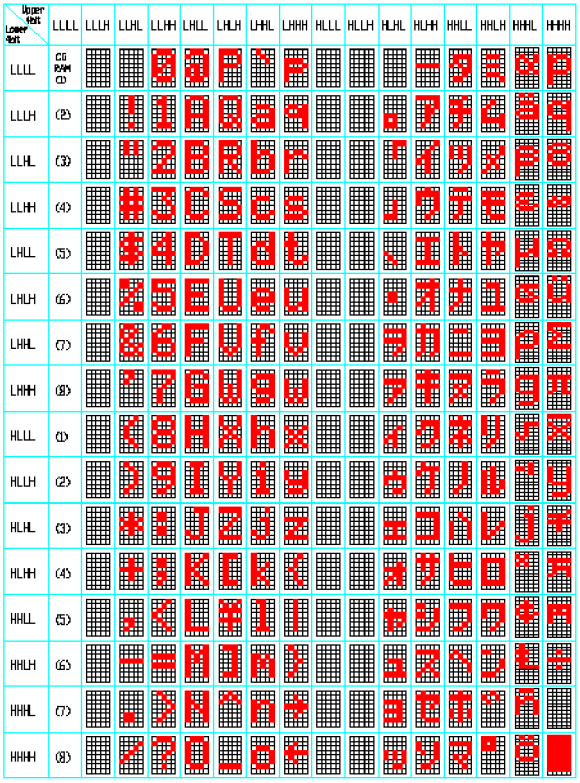


Fig. 3.2. Se muestra la tabla de todos los posibles caracteres que pueden aparecer en el display 16x2

Display 128x64

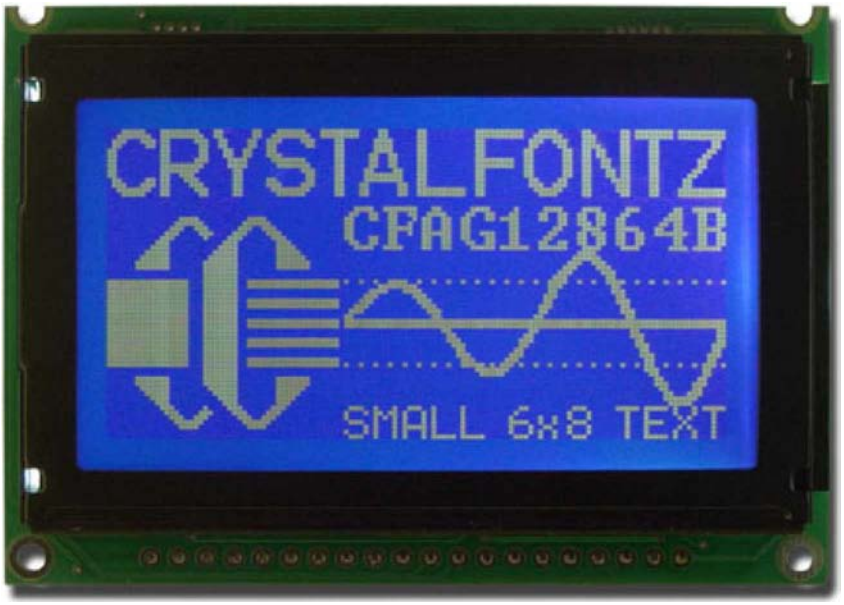


Fig. 3.3. Se muestra la parte frontal del display 128 puntos de largo por 64 puntos de alto

Explicación detallada de los pines

Pin 1 V_{DD} este Pin se conecta a Voltaje positivo a 5 Volts
Pin 2 Vss este Pin se conecta al Voltaje 0
Pin 3 Vo este Pin se conecta a un voltaje negativo variable que va de 0 a –7 Volts y que en algunos displays tienen un pin llamado Vee que genera el voltaje negativo y mediante una resistencia variable se regula el nivel de contraste
Pin 4 DB0 Pin no 0 para entrada de datos
Pin 5 DB1 Pin no 1 para entrada de datos
Pin 6 DB2 Pin no 2 para entrada de datos
Pin 7 DB3 Pin no 3 para entrada de datos
Pin 8 DB4 Pin no 4 para entrada de datos
Pin 9 DB5 Pin no 5 para entrada de datos
Pin 10 DB6 Pin no 6 para entrada de datos
Pin 11 DB7 Pin no 7 para entrada de datos
Pin 12 CS1 *Chip Select 1* este pin hace que los datos entrantes se escriban en el cuadrante no 1
Pin 13 CS2 *Chip Select 2* este pin hace que los datos entrantes se escriban en el cuadrante no 2

Nota: Combinación CS1 = 0 CS2 = 0 cuadrante 1 y 2 no reciben instrucciones ni datos, Combinación CS1 = 1 CS2 = 0 cuadrante 1 recibe instrucciones pero cuadrante no 2 no, Combinación CS1 = 0 CS2 = 1 cuadrante 2 recibe instrucciones pero cuadrante no 1 no recibe instrucciones, Combinación CS1 = 1 CS2 = 1 cuadrante 1 y 2 reciben las mismas instrucciones o datos.

Pin 14 RST Pin de reset cuando esta en 0 el display es reseteado
Pin 15 R/W READ o WRITE lee o escribe este pin en 1 lee datos desde el Display y cuando es 0 escribe datos en el Display
Pin 16 D/I Datos o instrucciones este pin es para seleccionar si son datos o instrucciones lo que queremos insertar al display
Pin 17 E Pin de enable. Este pin es para permitir la entrada de datos o instrucciones cuando este pin es 1 los datos en la entrada del display pueden cambiar cuando es el cambio de 1 a 0 lo que tenia en la entrada serán los datos o instrucciones los que se escriban cuando es el cambio de 0 a 1 otra vez pueden cambiar los datos
Pin 18 Vee este pin genera un voltaje negativo para controlar el contraste del display si se esta conectado a Vo
Pin 19 A Anodo para la luz azul amarilla o blanca de fondo si tiene el modelo del display (+)
Pin 20 K Katodo para la luz azul amarilla o blanca de fondo si tiene el modelo del display (-)

Explicación detallada de las instrucciones

Encendido de el Display (On or off)

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

Los puntos en el Display aparecen cuando R/W = 0, D/I = 0, DB7 = 0, DB7 = 0, y cuando D es 1 la pantalla aparece sin puntos hasta que D es 0 mientras puedes cargar toda la imagen y mostrarla hasta que este por completa. R/W y D/I deben estar en 0 y DB7 a DB0 como se muestra en el recuadro.

Inicio del display (Display Start Line)

R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	1	A	A	A	A	A	A

El pin R/W debe estar en 0 indicando que estamos escribiendo en el display, D/I debe estar en 0 que nos indica que es una instrucción, DB7 y DB6 en 1 es la instrucción para decirle al display que inicie en donde AAAAAA (en binario) inicie a dibujar el display los datos son tomados de la RAM.

Nota: CS1 y CS2 debe estar en 0 o 1 según en cual de los dos cuadrantes o los dos donde queremos que inicie el display cada cuadrante va de 0 a 63 posiciones posibles así de los dos son los 128 puntos.

Colocar la posición Y en el display (Set Y address)

R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	0	1	1	1	A	A	A

El pin R/W debe estar en 0 indicando que estamos escribiendo en el display, D/I debe estar en 0 que nos indica que es una instrucción, DB7 en 1 DB6 en 0 es la instrucción para decirle al display que inicie en donde AAA (en binario) inicie a dibujar de 8 posibles inicios en el display.

Nota: CS1 y CS2 debe estar en 0 o 1 según en cual de los dos o los dos cuadrantes en donde queremos que inicie el display cada cuadrante va de 0 a 7 posiciones posibles.

Colocar el cursor del display en la posición X (Set X address)

R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	A	A	A	A	A	A

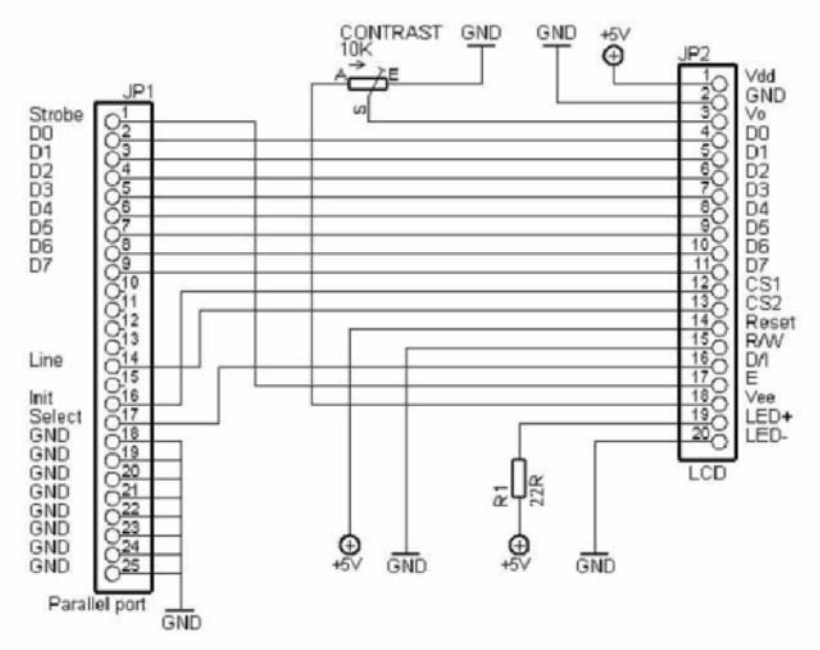
El pin R/W debe estar en 0 indicando que estamos escribiendo en el display, D/I debe estar en 0 que nos indica que es una instrucción, DB7 en 0 y DB6 en 1 es la instrucción para decirle al display que inicie en donde AAAAAA (en binario) inicie a dibujar el display los datos son tomados de la RAM.

Nota: CS1 y CS2 debe estar en 0 o 1 según en cual de los dos cuadrantes o los dos donde queremos que inicie el display cada cuadrante va de 0 a 63 posiciones posibles así de los dos son los 128 puntos.

Conexión del display al puerto paralelo.

Para controlar el display con el software LCD estudio hay que hacer el cable paralelo DB25 a las terminales del Display

Fig. 3.4. Se muestra las terminales de puerto DB25 hacia la conexión del display 128x64



Para el software necesitamos el Plug - In KS0108.

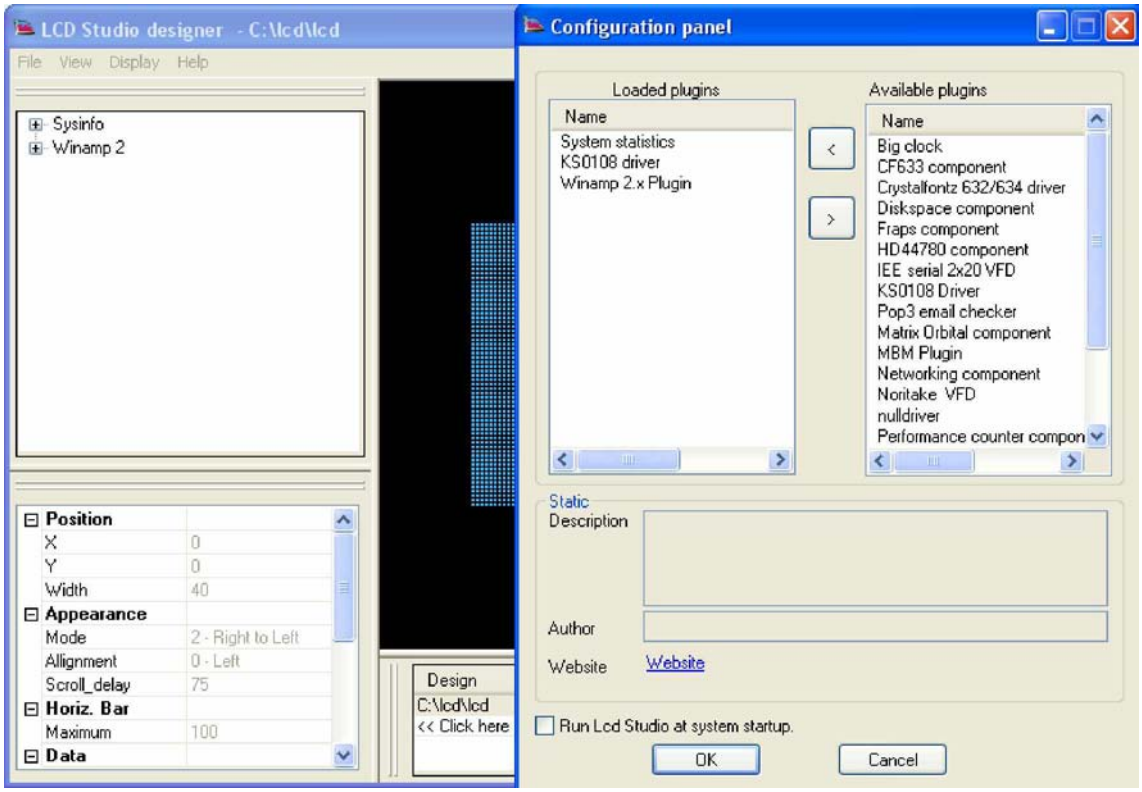


Fig. 3.5. Imagen del software que hace funcionar el display 128x64

Para inicializar el display tenemos que dar esta secuencia

Comando	D7	D6	D5	D4	D3	D2	D1	D0	EN	DI	RW	CS1	CS2
Encendido	0	0	1	1	1	1	1	1	1>0	0	0	0	0
Colocamos en la pagina 0 en los dos Cuadrantes	1	0	1	1	1	0	0	0	1>0	0	0	0	0
Colocamos en la posición 0 del eje x en los dos cuadrantes	0	1	0	0	0	0	0	0	1>0	0	0	0	0
Limpiamos una línea de 8 bits En el cuadrante 1 y avanza el cursor una posición	0	0	0	0	0	0	0	0	1>0	1	0	1	0

Simulador

Para este simulador daremos la secuencia anterior al software que se encuentra en esta pagina: http://www.geocities.com/SiliconValley/Circuit/8882/djlcdsim/djlcdsim.html
Encendido. En cuando Enable pasa de 1 a 0 el display se torna encendido, CS1 y CS2 están en bajo lo cual nos dice que la instrucción de encendido será en los dos cuadrantes

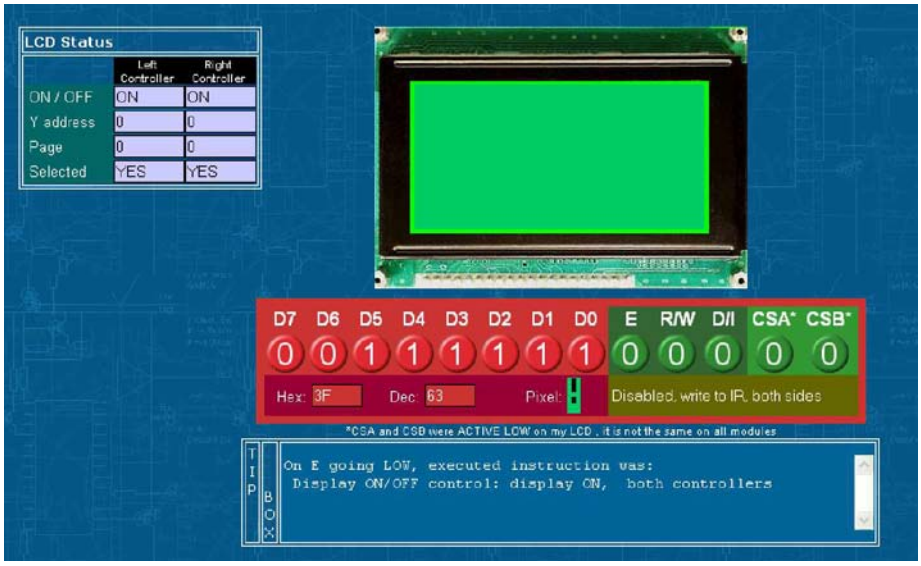


Fig. 3.6. Captura de la pantalla del software simulador del display 128x64.

Colocación en la pagina 1 de 8 posibles

El display tiene 8 barras de 8 bits en los dos cuadrantes, si queremos colocar ambas barras en la posición 0 CS1 y CS2 deben estar en bajo y D2 a D0 en bajo

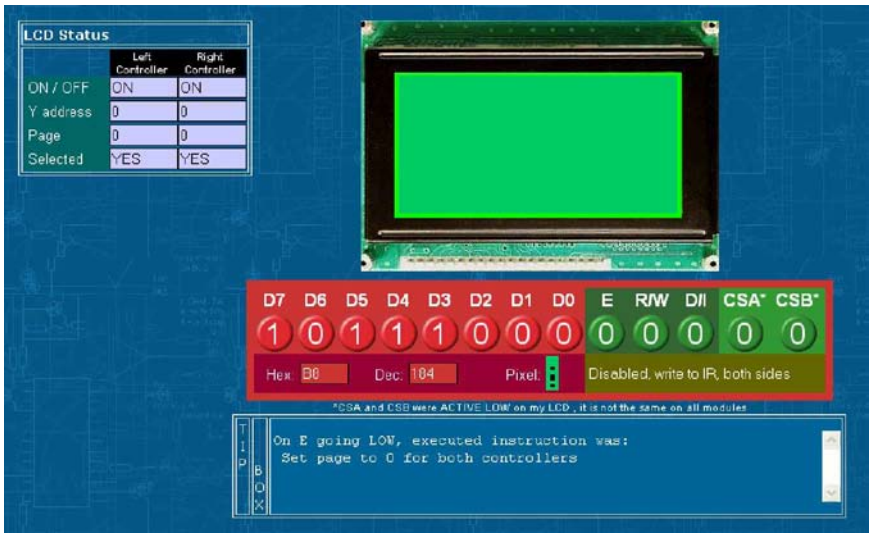


Fig. 3.7. Captura de la pantalla del software simulador del display 128x64.

Colocación en la posición 1 de 64 posibles en el eje Y.

Colocamos los datos como se muestran en la imagen y notamos que cuando se hace el cambio de Enable de 1 a 0 se coloca el cursor en la dirección 0 en Y. Como CS1 y CS2 están en bajo los dos cuadrantes son afectados

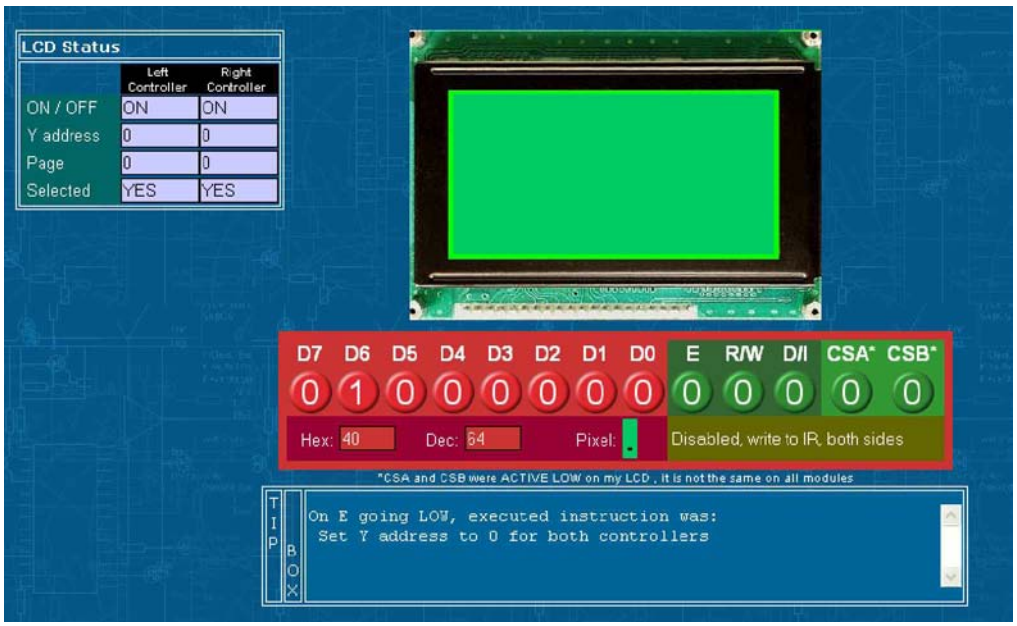
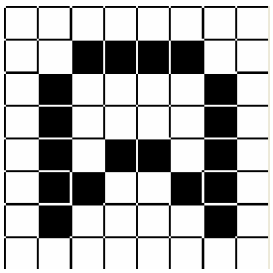


Fig. 3.7. Captura de la pantalla del software simulador del display 128x64. nota: solo D6 esta en alto

Iniciamos a pintar sobre el display

El display pinta barritas de 8 bits, como necesitamos el alfabeto y números hagamos una A en el cuadrante 1.



Primer barra de la letra A

Primer paso es seleccionar CS1 como 0 y CS2 como 1 esto hace que solo se pinte en uno de los dos cuadrantes

Segundo paso D7 a D0 deben estar en 1 o 0 según lo que queremos pintar como la letra A que queremos pintar inicia con todos en 0 pues ponemos todos en 0.

Tercer paso. Como ya no son instrucciones sino datos pasamos D/I a 1

Ahora si colocamos un 1 en enable y al momento de pasarlo a 0 se coloca el vacío y avanza una posición de las 64 posibles.

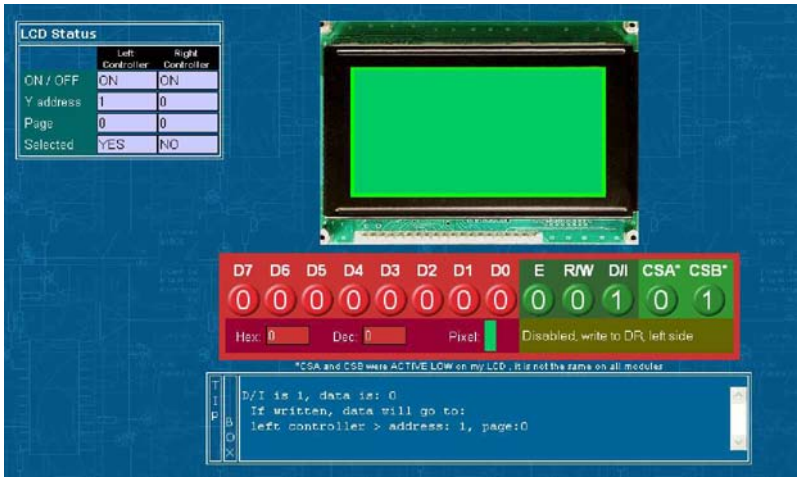


Fig. 3.8. Captura de la pantalla del software simulador del display 128x64. nota D/I y CSB están en alto.

Segunda barra de la letra A.

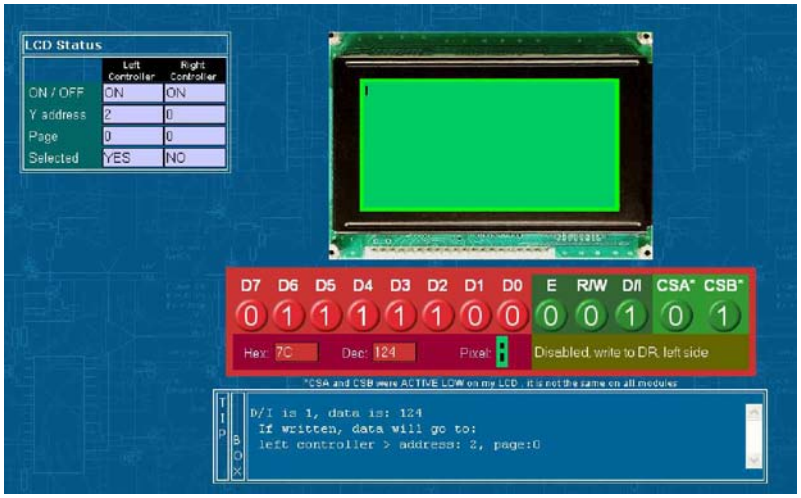


Fig. 3.9. Captura de la pantalla del software simulador del display 128x64. nota: D6, D5, D4, D3, D2, D/I y CSB están en alto.

Tercera barra de la letra A

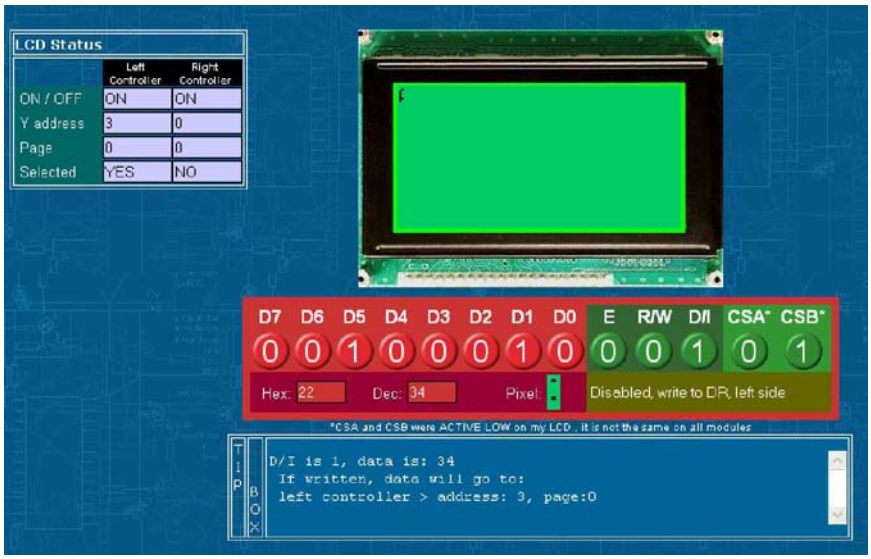


Fig. 3.10. Captura de la pantalla del software simulador del display 128x64. Nota: D5, D/I y CSB están en alto.

Cuarta barra de la letra A

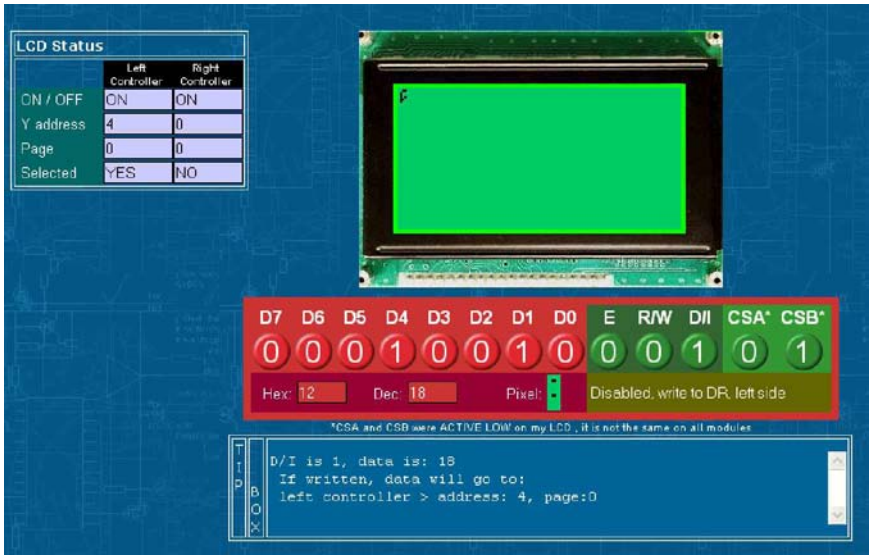


Fig. 3.11. Captura de la pantalla del software simulador del display 128x64. Nota: D6, D5, D4, D3, D2, D/I y CSB están en alto.

Quinta barra de la letra A

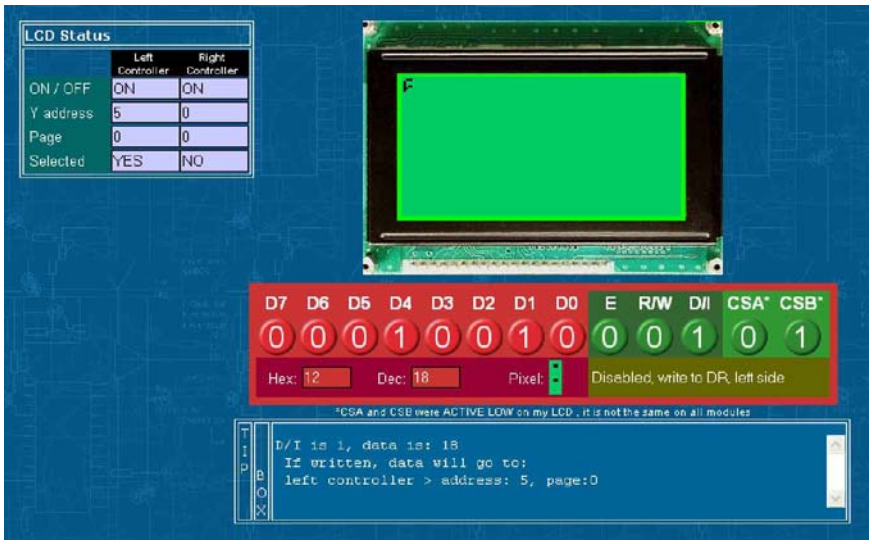


Fig. 3.12. Captura de la pantalla del software simulador del display 128x64. Nota: D4, D1, D/I y CSB están en alto.

Sexta barra de la letra A

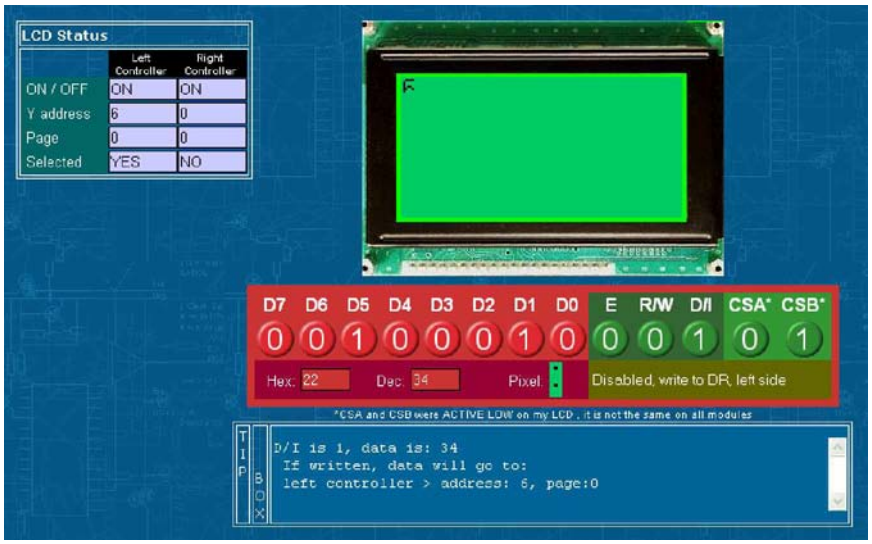


Fig. 3.13. Captura de la pantalla del software simulador del display 128x64. Nota: D5, D2, D/I y CSB están en alto.

Séptima barra de la letra A

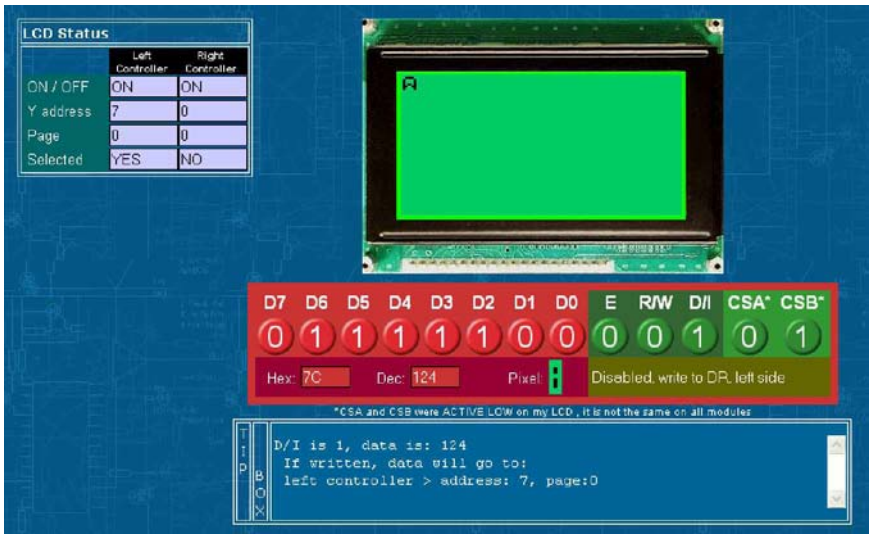


Fig. 3.14. Captura de la pantalla del software simulador del display 128x64. Nota: D7, D6, D, 5, D4, D3, D2, D/I y CSB están en alto.

Octava barra de la letra A

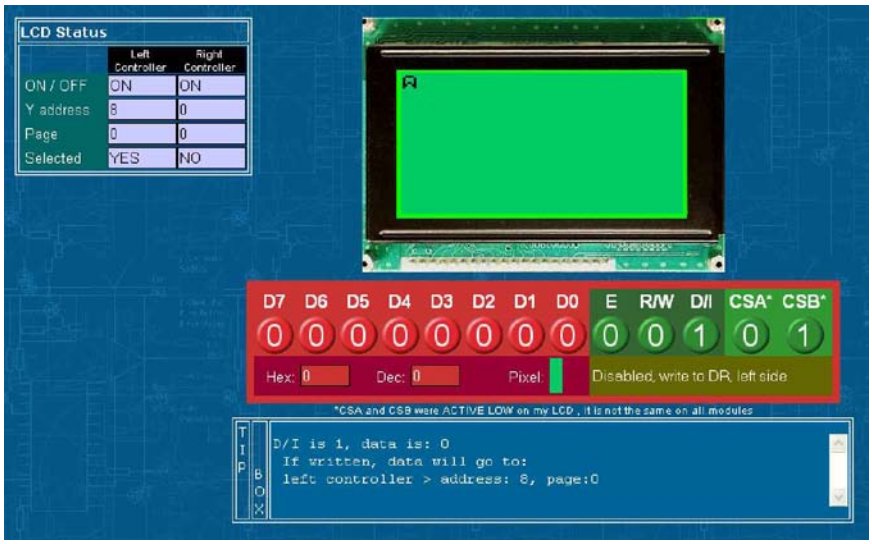
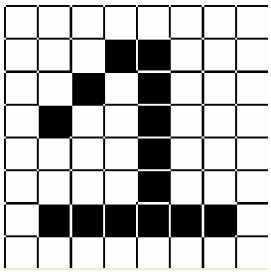


Fig. 3.15. Captura de la pantalla del software simulador del display 128x64. Nota: Solo D/I y CSB están en alto.

Hasta este punto vemos que avanzamos 8 direcciones en la pagina 0

El display funciona por barras 8 barras de 8 bits hasta 64 espacios por 2 paginas así se logran los 128 puntos por 64 así que ahora comenzamos a pintar un numero 1 en la pagina 0 del cuadrante 2. Solo por poner un ejemplo.



Paso no 1. Primera barra en el segundo cuadrante de la primera línea

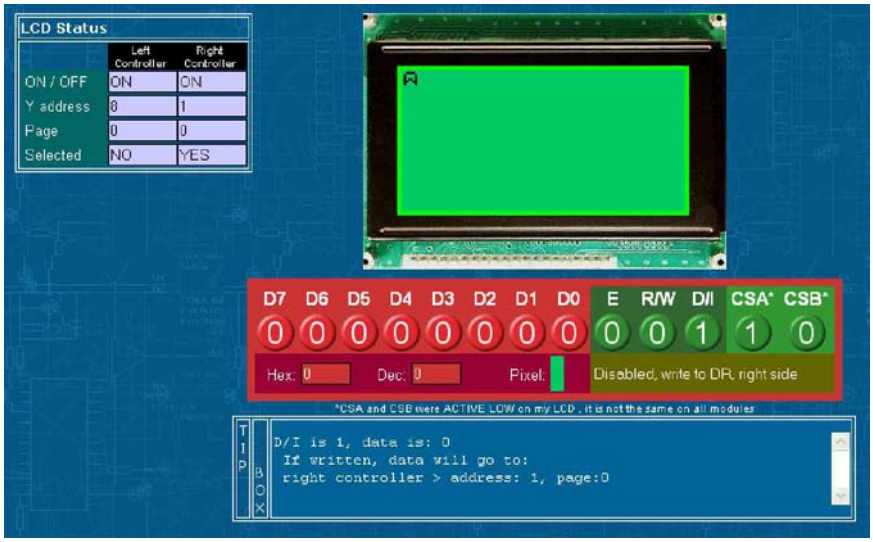


Fig. 3.16. Captura de la pantalla del software simulador del display 128x64. Nota: Solo D/I y CSA están en alto.

Paso no 2

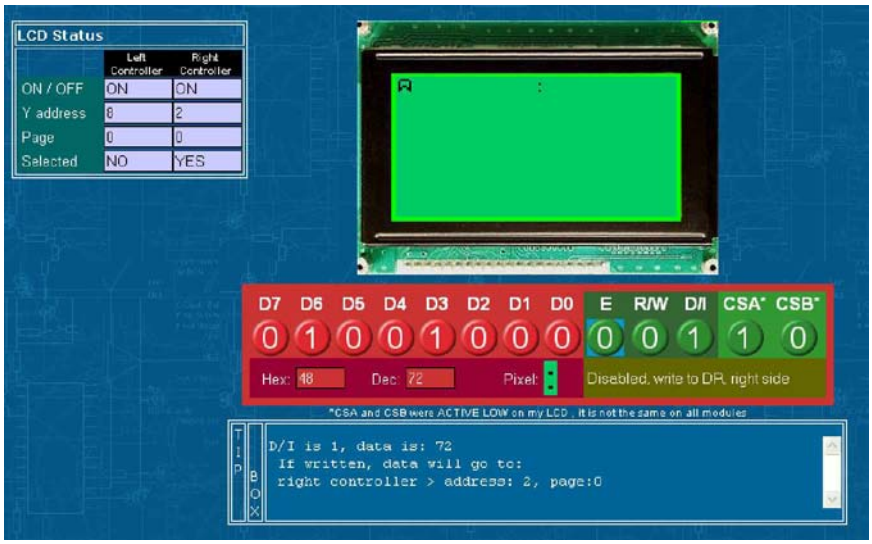


Fig. 3.17. Captura de la pantalla del software simulador del display 128x64. Nota: D6, D3, D/I y CSA están en alto.

Paso no 3

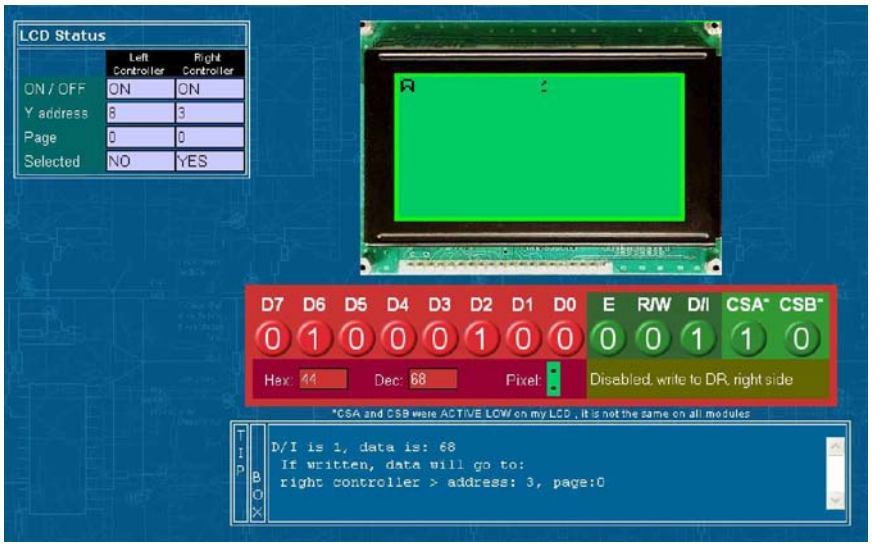


Fig. 3.18. Captura de la pantalla del software simulador del display 128x64. Nota: D6, D2, D/I y CSA están en alto.

Paso no 4

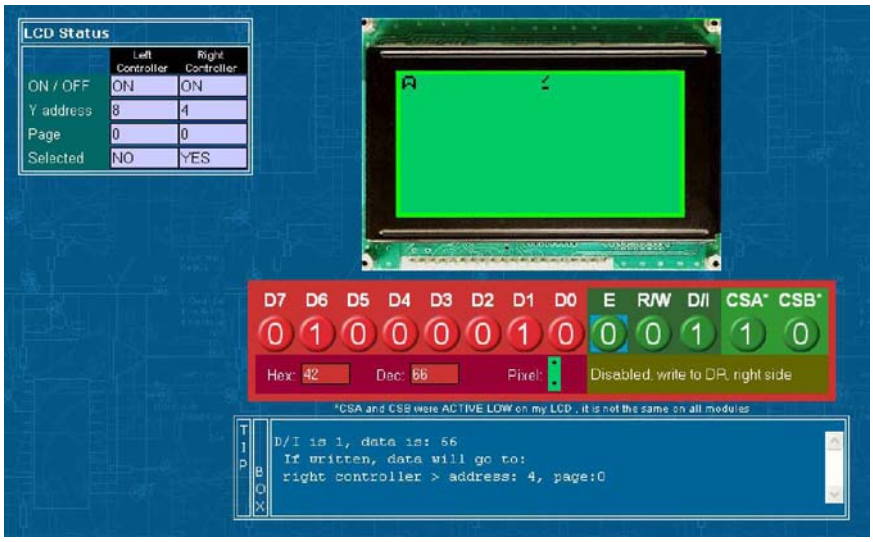


Fig. 3.19. Captura de la pantalla del software simulador del display 128x64. Nota: D6, D1, D/I y CSA están en alto.

Paso no 5

Fig. 3.20. Captura de la pantalla del software simulador del display 128x64. Nota: D6, D5, D4, D3, D2, D1, D/I y CSA están en alto.

Paso no 6

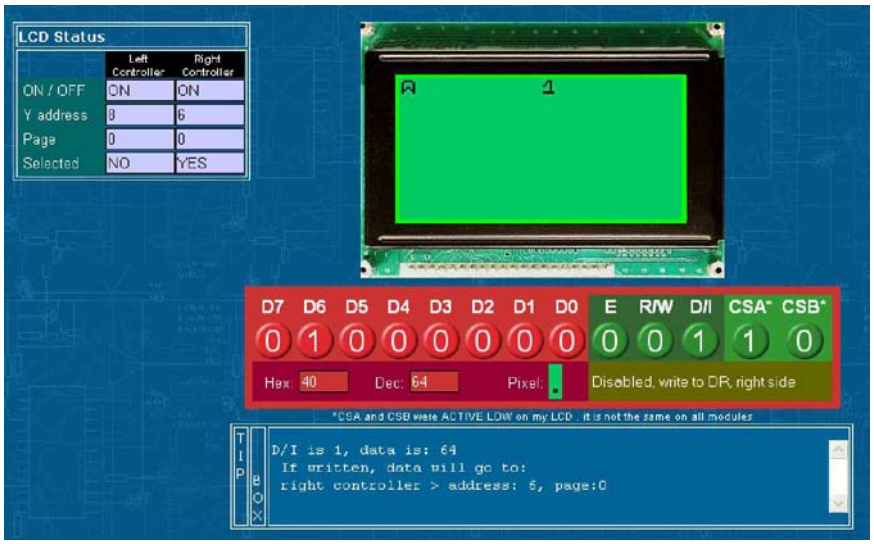


Fig. 3.21. Captura de la pantalla del software simulador del display 128x64. Nota: Solo D6, D/I y CSA están en alto.

Paso no 7

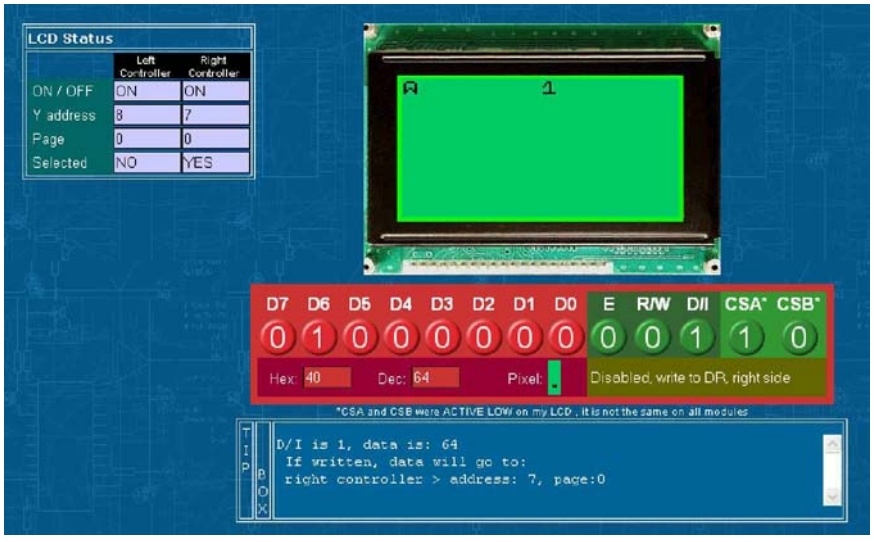


Fig. 3.22. Captura de la pantalla del software simulador del display 128x64. Nota: solo D6, D/I y CSA están en alto.

Paso no 8

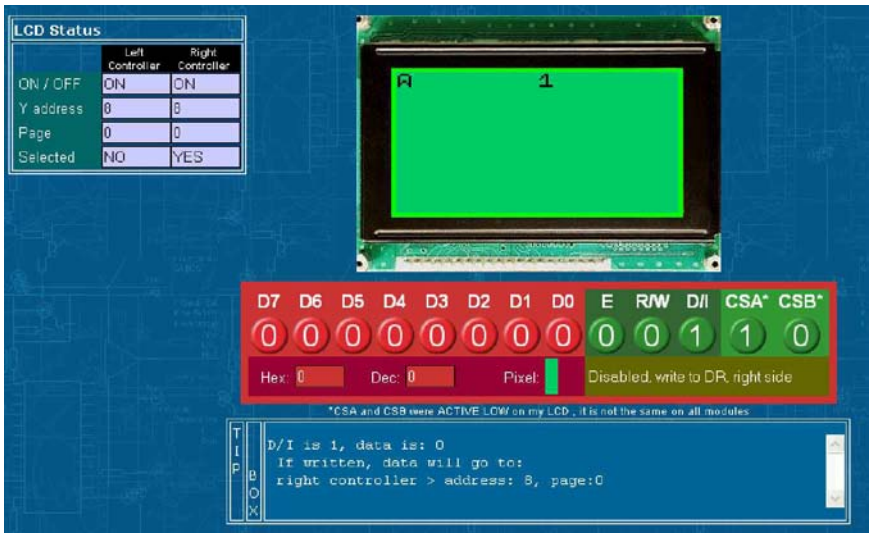


Fig. 3.23. Captura de la pantalla del software simulador del display 128x64. Nota: solo D/I y CSA están en alto.

Así logramos hacer dibujos caracteres o lo que queramos pintar en el display.

Control remoto IV

Control por Infrarrojo

Recepción

Vamos a comenzar por definir y comentar la parte de la recepción de la señal infrarroja ya que, por un lado, es sumamente sencilla de conectar a un microcontrolador y, por otro, es la que nos va a obligar a diseñar y ajustar los circuitos que necesitamos en la parte de la emisión.

Para la recepción vamos a utilizar un dispositivo que unifica en el mismo encapsulado el receptor de luz infrarroja, una lente y toda la lógica necesaria para distinguir señales moduladas a una determinada frecuencia. Concretamente, en este montaje utilizaremos los receptores IS1U60 de Sharp que se activan cuando reciben una luz infrarroja modulada a una frecuencia de 38 kHz (el haz infrarrojo se apaga y enciende 38000 veces por segundo). Esto los hace compatibles con un gran número de mandos a distancia de electrodomésticos.

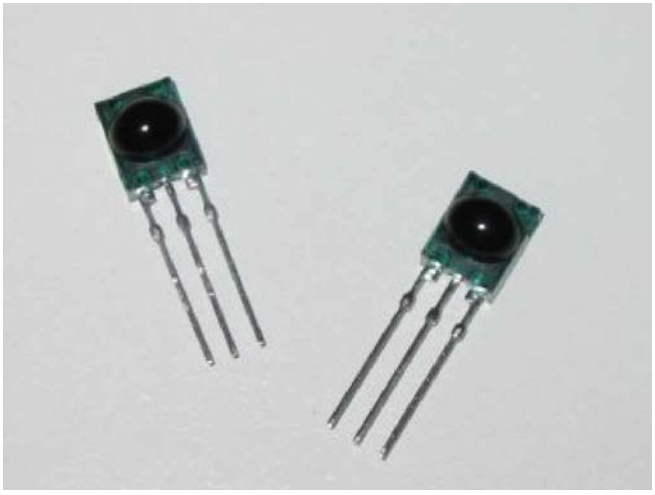


Fig. 4.1. Par de receptores infrarrojos se hace referencia por diversos nombres dependiendo del fabricante.

Para la recepción se ha utilizado un circuito receptor de infrarrojos como es el IS1U60 de Sharp, que en el mismo integrado engloba el receptor infrarrojo, lentes para mejorar la recepción y la lógica necesaria. Desde el punto de vista de su utilización, este dispositivo lo podemos ver como una caja negra que una vez alimentada, activa o desactiva una salida cuando recibe una señal infrarroja modulada a 38 kHz.

Emisión.

Programa en Basic para generar los 38 Khz.

La suma de los dos retardos 12 milisegundos y 13 milisegundos en el programa nos da como resultado aproximadamente los 38 khz.

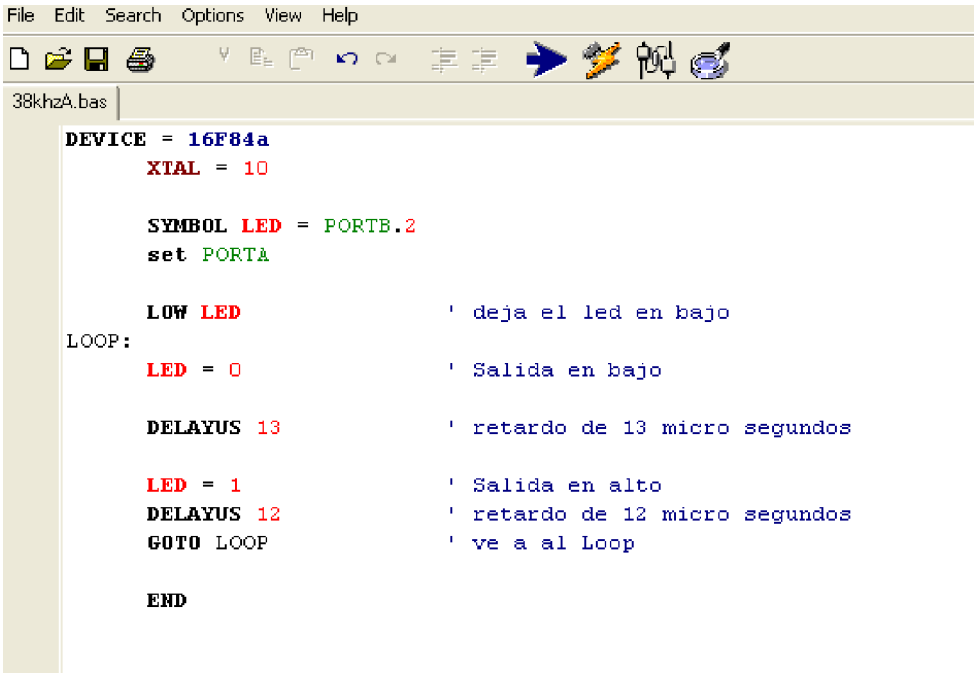


Fig. 4.2. Imagen capturada de la pantalla donde se muestra el programa con la sintaxis correcta.

MC145026 MC145027 MC145028

Este circuito integrado MC145026 solo se usa en el transmisor y sirve para generar una señal codificada que será aplicada en la etapa correspondiente para ser emitida.

El circuito integrado MC145027 y MC145028 se utilizan solamente en el receptor.

Fig. 4.3.

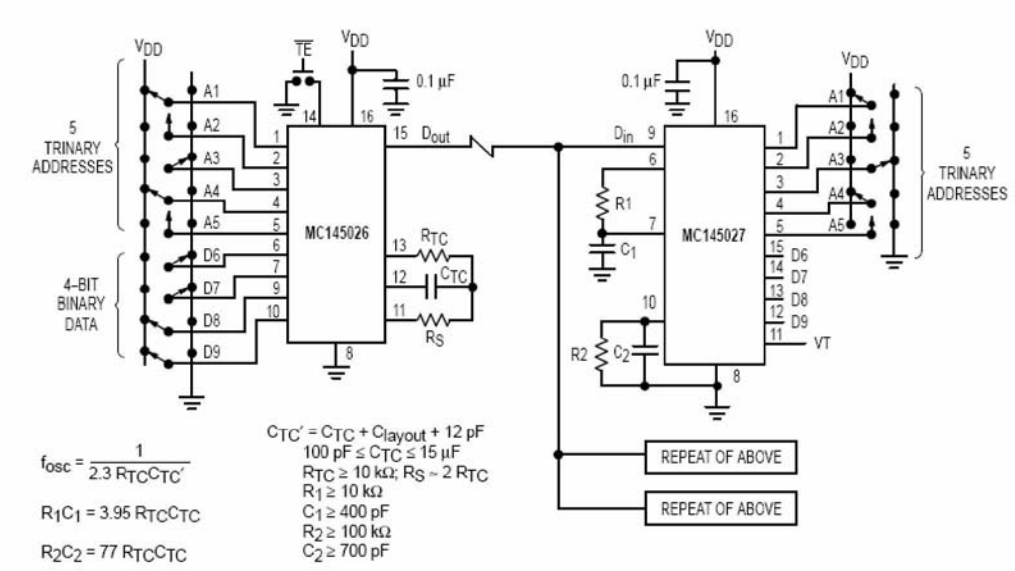


Imagen tomada de las hojas de datos de los chips MC145026, MC145027 y MC145028 respectivamente.

El circuito integrado MC145026

Este circuito solo se usa en el transmisor y sirve para generar una señal codificada que será aplica en la etapa correspondiente para ser emitida. En la salida del integrado pin 15 se obtiene una señal codificada, formada por una secuencia de palabras o trenes de pulsos que se emiten constantemente mientras se tenga conectado a masa, mediante un pulsador el pin 14 es el pin de habilitación. Como se observa en la figura 1, el integrado está conectado permanentemente, sin embargo en esta citación no consume corriente y solo lo hace cuando entra en acción (pin 14) a masa. Cada palabra contiene una secuencia de 9 bits y cada bit esta formado por dos pulsos.

El circuito integrado MC145027

Este circuito integrado solo se usa en el receptor. Este circuito usa un comparador lógico que separan las señales que corresponden a los datos recibidos, que en este caso pertenecen a los pines 6,7,9 y 10 del integrado 145026, en el caso de usar 5026-5027 solo disponemos de 5 pines para codificar en ambos integrados y 4 pines para datos

Cálculos para el codificador del Infrarrojo

$$F_{osc} = \frac{1}{(2.3)(RTC)(CTC')}$$

$F_{osc} = 1$ en kHz

$CTC' = 4.7_{\mu f} + 0.02_{\mu f} = 4.72_{\mu f}$

$1_{kHz} = (2.3)(RTC)(4.72x10^{-6})$

$$RTC = \frac{1}{(4.72x10^{-6})(2.3)} = 92.11495947_{k\Omega}$$

Valor Comercial = $100k_{k\Omega}$

$$\text{Khz} \cdot F_{osc} = \frac{1}{(2.3)(100x10^{3})(4.72_{\mu f})}$$

$F_{osc} = 921.1495947x10^{-3} \cdot 1x10^{3}$

$F_{osc} = 921.1495947Hz$

$Rs = 2x92.11495957k = 2x92k = 184k\Omega$

$Rs = 2xRTC = 2x100k = 200k\Omega$

Valor Comercial = 200kΩ

$R1 \geq 10k\Omega$

$R1 = K\Omega$

$$R1 = \frac{3,95 \cdot RTC \cdot CTC}{C1} = \frac{3.95 \cdot 100 \cdot 4.7x10^{-6}}{10x10^{-6}} = 185.650 * 1000$$

$R1 = 185.650k\Omega$

Valor Comercial 180k or 220k

R1=180k

$R1 \cdot C1 = 3.95 \cdot RTC \cdot CTC$

$R1 \cdot C1 = 3.95 \cdot 100k \cdot 10.0\mu f = 1.8565$

$3.95 \cdot RTC \cdot CTC = 3.95 \cdot 100k \cdot 4.7\mu f = 1.8565$

$$R2 = \frac{77 \cdot RTC \cdot CTC}{C2} = \frac{77 \cdot 100k \cdot 10\mu f}{0.47\mu f}$$

$R2 = 163.8297872k$

$R2 = 180k$

Hasta ahora tenemos la señal moduladora codificada pero para la señal portadora programamos un Microcontrolador PIC16F84a para generar 38 Khz. y con una compuerta AND se multiplica las dos señales. Y así tenemos la señal moduladora y portadora juntas.

Señal portadora y moduladora

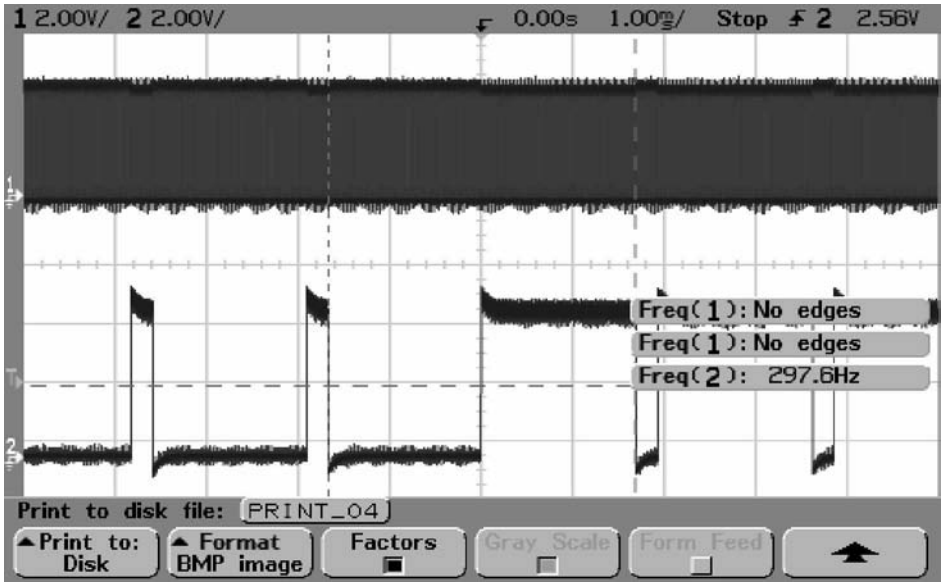


Fig. 4.4. En esta imagen se muestra la señal portadora de 38 khz en la parte superior y en la parte inferior la señal codificada (Imagen tomada del osciloscopio Agilent del laboratorio de IME)

Señal moduladora y portadora juntas

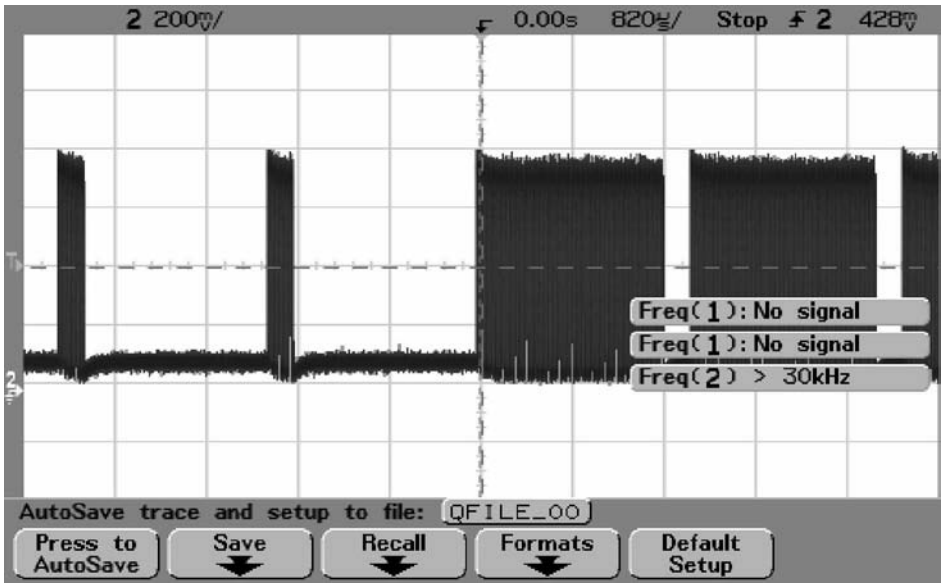


Fig. 4.5. En esta imagen se muestra la señal portadora de 38 khz y la señal codificada juntas (Imagen tomada del osciloscopio Agilent del laboratorio de IME)

Señal moduladora y portadora filtrada por el receptor activo en bajo

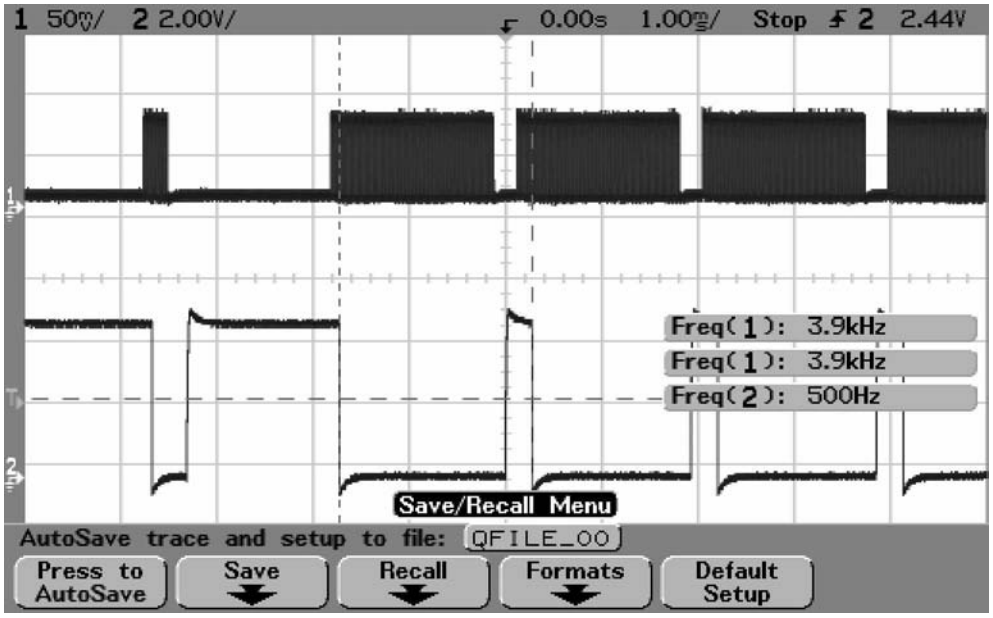


Fig. 4.6. En esta imagen en la parte superior se muestra la señal recibida por el censor Samsung pero este censor por hojas de datos se activa en bajo en el mismo circuito se coloca un inversor para que recibamos bien la señal. Señal inferior invertida (Imagen tomada del osciloscopio Agilent del laboratorio de IME)

Señal moduladora y portadora filtrada por el receptor activo en bajo con un inversor para recibir bien la señal

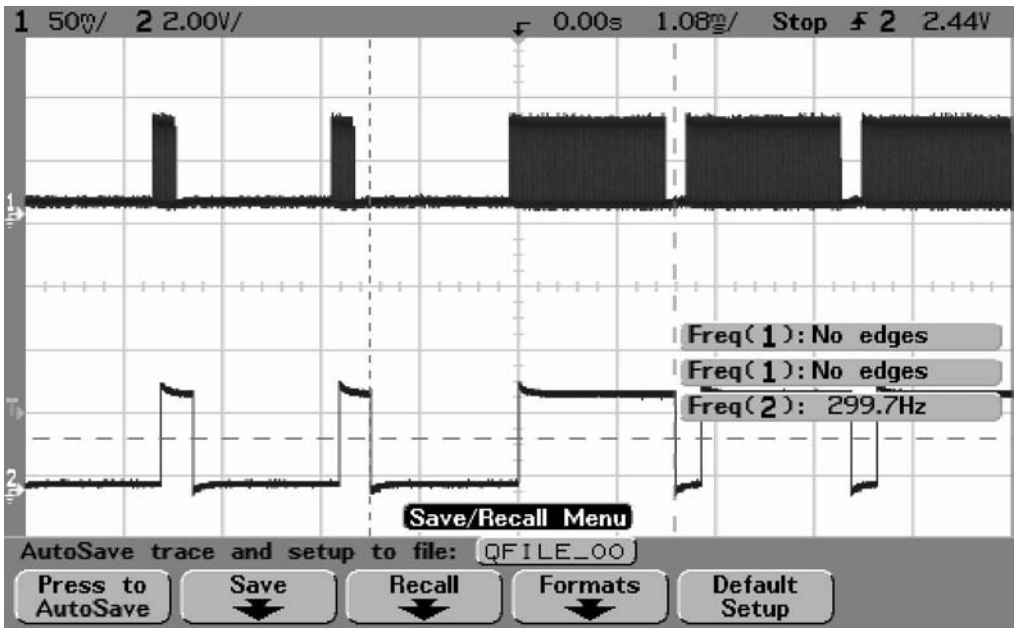


Fig. 4.7. En esta imagen en la parte superior se muestra la señal emitida desde el led infrarrojo. En la parte inferior se muestra la señal ya filtrada por el censor sansung. (Imagen tomada del osciloscopio Agilent del laboratorio de IME).

Microcontroladores PIC Serie 18FXX V

Características.

A continuación describiremos algunos de los Microcontroladores PIC para mayor entendimiento de lo que se puede hacer y no con sus respectivas características de cada uno de los modelos mas comunes.

DATOS PRINCIPALES DEL MICROCONTROLADOR PIC16F84a

Sus principales características son dos puertos A y B, el puerto A esta compuesto por 5 pines del pin 0 al 4, el pin llamado RA4/T0CKI es utilizado para un reloj externo o una salida, cuando configuramos este pin como salida tenemos que colocarle una resistencia a tierra, el puerto B tiene la característica que se puede configurar como entrada y salidas, pero cuando están configuradas como entrada se le puede añadir la característica de resistencia de pull – up, esto hace que se dispare a 5 voltios cuando no es 0.

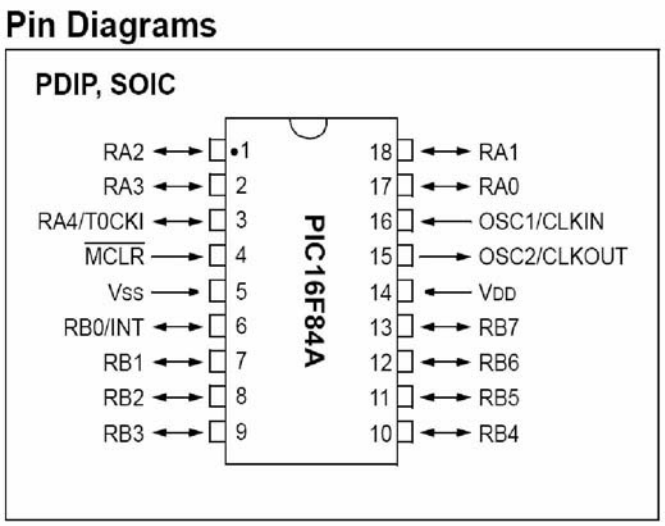
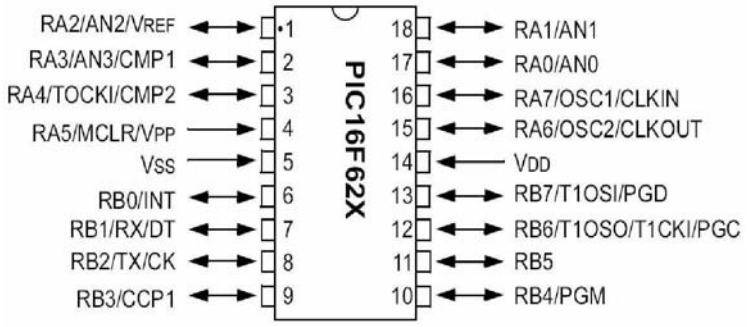


Fig. 5.1. Imagen donde se muestran las terminales del microcontrolador PIC16F84A

DATOS PRINCIPALES DEL MICROCONTROLADOR PIC16F642

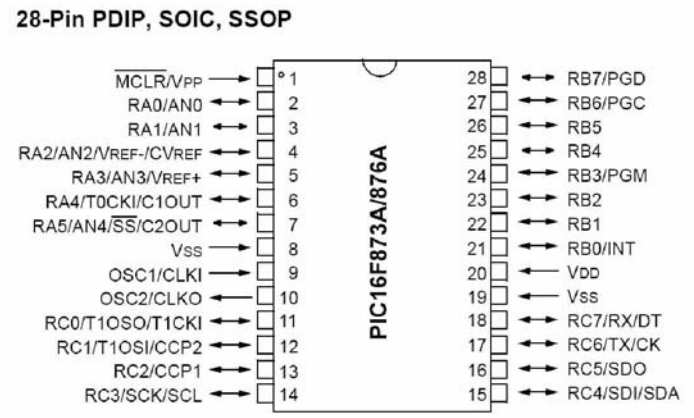
Este PIC es parecido al 16F84 pero con la diferencia principal entre muchas que tiene por hardware una interrupción llamada HPWM, modulación de ancho de pulsos.



5.2. Imagen donde se muestran las terminales del microcontrolador PIC16F62X

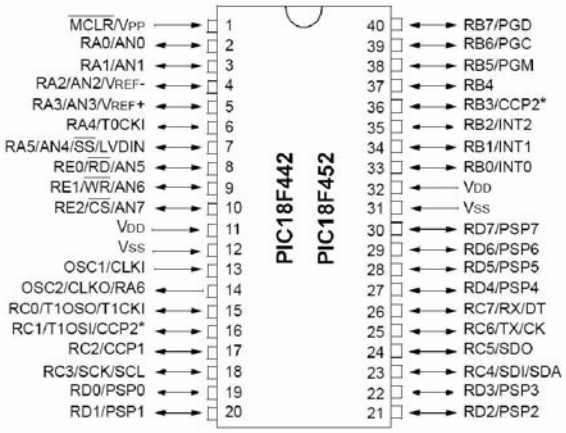
DATOS PRINCIPALES DEL MICROCONTROLADOR PIC16F876

Este microcontrolador aparte de tener HPWM y las características de los otros PICs, tiene 6 convertidores análogos a digital en el puerto A.



5.3. Imagen donde se muestran las terminales del microcontrolador PIC16F876A

DATOS PRINCIPALES DEL MICROCONTROLADOR PIC18F452



Este PIC tiene 8 convertidores análogo digital en el puerto A y E, 5 en el puerto A y 3 en el puerto E aparte de todas las características anteriores tiene doble HPWM, entre otras características.

5.4. Imagen donde se muestran las terminales del microcontrolador PIC16F452

Lenguaje ensamblador para PIC VI

Técnicas de programación

Con esto solo pretendo introducir al lector en el mundo de la programación de microcontroladores PIC de forma práctica y sencilla. Doy por supuestos unos conocimientos muy básicos sobre electrónica digital. Hablaremos de Instrucciones, registros de memoria RAM, memoria EEPROM (un tipo de ROM), de niveles lógicos "0" o "1" y cosas por el estilo.

El PIC16F84 es un microcontrolador, una especie de "ordenador en miniatura" (con muchas comillas) que podremos programar. En su interior posee un microprocesador, una memoria RAM (volátil) donde guardaremos las variables, una memoria EEPROM (no volátil) donde guardaremos nuestro programa, un Timer o contador que nos facilitará algunas tareas, y alguna cosilla mas…

Algunas características más representativas son:

Opera a una frecuencia máxima de 10 MHz.
1Kbyte de memoria EEPROM para nuestro programa
68 bytes (de 8 bits) de memoria RAM
64 bytes de memoria EEPROM para datos (no volátiles)
Solo 35 instrucciones
13 pines de entrada/salida (un puerto de 8 bits + otro de 5 bits)
Timer/contador de 8 bits.

Descripción de sus pines

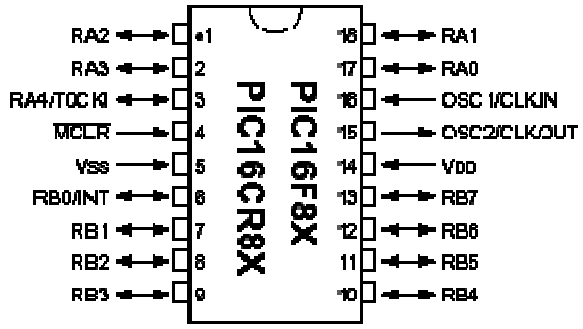


Fig. 5.5. Imagen donde se muestran las terminales del microcontrolador PIC16F84A.

RA0, RA1, RA2, RA3 y RA4: son los pines del puerto A.

RB0, RB1, RB2, RB3, RB4, RB5, RB6 y RB7: son los pines del puerto B.

MCLR: Pin de reseteo del pic, cuando se pone a "0" el pic se resetea.

Vdd y Vss: pines de alimentación (Vdd 5V y Vss a masa).

OSC1/CLKIN y OSC2/CLKOUT: son para el oscilador Los tipos de osciladores mas usados son el XT (cristal de cuarzo) y el RC (resistencia y condensador) el modo de conexionado es el siguiente.

Oscilador XT	Oscilador RC
C1=C2=33pF Cristal ≤ 4MHz	C1 alrededor de 20pF 5KΩ ≤ R1 ≤100KΩ

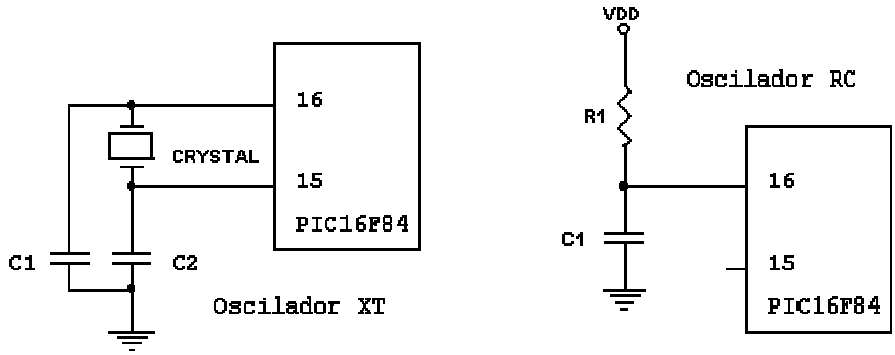


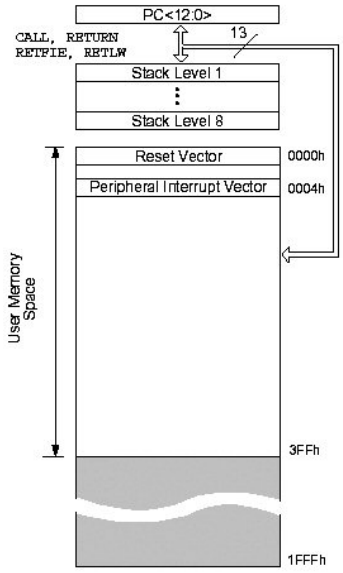
Fig. 6.1. Imagen donde se muestran los tipos de osciladores del microcontrolador PIC16F84A.

Organización de la memoria

En primer lugar tenemos que distinguir claramente entre tres tipos de memoria:

Una: la memoria EEPROM donde almacenaremos el programa que haremos, esta memoria solo podrá ser leída por el PIC (el PIC va leyendo las instrucciones del programa almacenado en esta memoria y las va ejecutando). Al apagar el pic esta memoria no se borra.

Dos: la memoria RAM en cuyos registros se irán almacenando los valores de las variables que nosotros queramos y cuando nosotros queramos (por programa), al apagar el pic esta memoria se borra.



Tres: la memoria EEPROM para datos, es un espacio de memoria EEPROM en la que se pueden guardar variables que queremos conservar aunque se apague el pic. No se tratará aquí por ser una memoria más difícil de emplear.

La memoria EEPROM o memoria de programa

El espacio marcado como "User memory Space" es el espacio de memoria donde irá nuestro programa, comprende las direcciones de memoria desde la 0000h hasta la 3FFh (3FFh en decimal es 1023, mas la dirección 0000h hacen 1024 direcciones, es decir, 1Kbyte).
"Reset Vector" es la primera dirección a la que se dirige el PIC al encenderlo o al resetearlo.

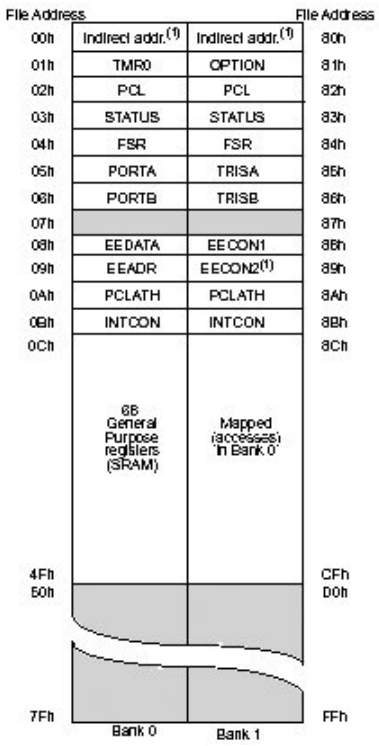
Fig. 6.2. Imagen donde se muestra el mapa de la memoria del PIC16F84a.

"PC" y los "Stack Level" son empleados por el PIC y nosotros no tenemos acceso a ellos.

La memoria RAM

La memoria RAM no solo se usa para almacenar nuestras variables, también se almacenan una serie de registros que configuran y controlan el PIC. Podemos observar en la imagen que esta memoria esta dividida en dos bancos, el banco 0 y el banco 1, antes de acceder a un registro de esta memoria tenemos que preguntarnos en que banco estamos, generalmente se trabaja en el banco 0, ya veremos mas adelante como cambiamos de banco de memoria. Fijándonos en el banco 0, las direcciones desde la 00h hasta la 0Bh están ocupadas por registros del pic, por lo que tendremos que empezar a guardar nuestras variables a partir de la dirección 0Ch. Podremos acceder al resto de registros para cambiar la configuración o el estado del pic.

Faltaría añadir a este cuadro el registro mas utilizado de todos, el acumulador (W) o registro de trabajo. No se trata de un registro propiamente dicho ya que no tiene dirección pero se usa constantemente para mover datos y dar valore a las variables (registros). Por ejemplo, si queremos copiar la información del registro 0Ch en el registro 0Dh no podremos hacerlo directamente, deberemos usar una instrucción para cargar el valor del registro 0Ch en el acumulador W y después otra instrucción para cargar el valor del acumulador en el registro 0Bh



6.3. Imagen donde se vuelve a mostrar el mapa de memoria del Microcontrolador Pic16f84a.

Registros internos

A continuación se explican todos los registros de configuración y control de la memoria RAM. Estos registros se usan para controlar los pines del pic, consultar los resultados de las operaciones de la ALU (unidad aritmética lógica), cambiar de banco de memoria... entre otras cosas.

En BANCO "0"

INDF (direccionamiento indirecto): Dirección 00h, sirve para ver el dato de la dirección a la que apunta el registro FSR (dir. 04h) que veremos mas adelante.

TMR0 (Timer/contador): Dirección 01h, Aquí se puede ver el valor en tiempo real del Timer/contador. También se puede introducir un valor y alterar así el conteo. Este conteo puede ser interno (cuenta ciclos de reloj) o externo (cuneta impulsos introducidos por RA4).

PCL (Parte baja del contador de programa): Dirección 02h, Modificando este registro se modifica el contador de programa, este contador de programa es el que señala al pic en que dirección (de EEPROM) tiene que leer la siguiente instrucción. Esto se utiliza mucho para consultar tablas (ya veremos mas adelante)

STATUS: Dirección 03h, este es uno de los registros mas importantes y el que mas vas a utilizar. Hay que analizar el funcionamiento de este registro bit a bit:

CARRY, Dirección STATUS,0 (bit 0): bit de desbordamiento. Este bit se pone a "1" cuando la operación anterior ha rebasado la capacidad de un byte. Por ejemplo, si sumo dos números y el resultado no cabe en 8 bit el CARRY se pone a "1", Pasa lo mismo cuando resto dos números y el resultado es un número negativo. Se puede usar para saber si un número es mayor que otro (restándolos, si hay acarreo es que el segundo era mayor que el primero). Una vez que este bit se pone a "1" no se baja solo (a"0"), hay que hacerlo por programa si queremos volverlo a utilizar.

DC (digit carry), Dirección STATUS,1 (bit 1): lo mismo que el anterior pero esta vez nos avisa si el número no cabe en cuatro bits.

Z (zero), Dirección STATUS,2 (bit 2): Se pone a "1" si la operación anterior ha sido cero. Y pasa a "0" si la operación anterior no ha sido cero. Se usa para comprobar la igualdad entre dos números (restándolos, si el resultado es cero ambos números son iguales)

PD (Power - Down bit), Dirección STATUS,3 (bit3) se pone a "0" después de ejecutar la instrucción SLEEP*, se pone a "1" después de ejecutar la instrucción CLRWDT* o después de un power-up*.

TO (Timer Up), Dirección STATUS,4 (bit4) se pone a "0" cuando se acaba el tiempo del WATCHDOG*, Se pone a "1" después de ejecutar las instrucciones, CLRWDT* o SLEEP* o después de un power-up*.

RP0 y RP1 (selección de banco), Dirección STATUS,5 y STATUS,6. Como el PIC16F84 solo tiene dos bancos de memoria el RP1 no se usa para nada, la selección del banco se hace mediante RP0 (STATUS,5), si está a "0" nos encontramos en el banco 0, y si está a "1" nos encontramos en el banco 1.

IRP, Dirección STATUS,7, En este PIC no se usa para nada.

FSR (Puntero), Dirección 04h, se usa para direccionamiento indirecto en combinación con el registro INDF (dir. 00h): se carga la dirección del registro que queremos leer indirectamente en FSR y se lee el contenido de dicho registro en INDF.

PORTA (Puerto A), Dirección 05h. Con este registro se puede ver o modificar el estado de los pines del puerto A (RA0 - RA4). Si un bit de este registro está a "1" también lo estará el pin correspondiente a ese bit. El que un pin esté a "1" quiere decir que su tensión es de 5V, si está a "0" su tensión es 0V.
Correspondencia:

RA0 ==> PORTA,0

RA1 ==> PORTA,1

RA2 ==> PORTA,2

RA3 ==> PORTA,3

RA4 ==> PORTA,4

PORTB (Puerto B), Dirección 06h igual que PORTA pero con el puerto B
Correspondencia:

RB0 ==> PORTB,0

RB1 ==> PORTB,1

RB2 ==> PORTB,2

RB3 ==> PORTB,3

RB4 ==> PORTB,4

RB5 ==> PORTB,5

RB6 ==> PORTB,6

RB7 ==> PORTB,7

Dirección 07h, No utilizada por este PIC.

EEDATA, Dirección 08h. En este registro se pone el dato que se quiere grabar en la EEPROM de datos

EEADR, Dirección 09h. En este registro se pone la dirección de la EEPROM de datos donde queremos almacenar el contenido de EEDATA

PCLATH, Dirección 0Ah. Modifica la parte alta del contador de programa (PC), el contador de programa se compone de 13 bits, los 8 bits de menor peso se pueden modificar con PCL (dir. 02h) y los 5 bits de mayor peso se pueden modificar con PCLATH

INTCON (controla las interrupciones), Dirección 0Bh. Se estudia bit a bit:

RBIF (Flag de interrupción por cambio de PORTB) Dirección INTCON,0 (bit 0) se pone a "1" cuando alguno de los pines RB4, RB5, RB6, o RB7 cambia su estado. Una vez que está a "1" no pasa a "0" por si mismo: hay que ponerlo a cero por programa.

INTF (Flag de interrupción de RB0) Dirección INTCON,1. Si está a "1" es que ha ocurrido una interrupción por RB0, si está a "0" es que dicha interrupción no ha ocurrido. Este bit es una copia de RB0.

TOIF (Bandera de interrupción por desbordamiento de TMR0) Dirección INTCON, 2.

Cuando TMR0 se desborda esta bandera avisa poniéndose a "1". Colocar a "0" por programa.

RBIE (Habilita la interrupción por cambio de PORTB) Dirección INTCON, 3. Si está a "1" las interrupciones por cambio de PORTB son posibles.

INTE (Habilita la interrupción por RB0) Dirección INTCON,4. Si lo ponemos a "1" la interrupción por RB0 es posible

TOIE (Habilita la interrupción por desbordamiento de TMR0) Dirección INTCON,5. Si este bit esta a "1" la interrupción por desbordamiento de TMR0 es posible.

EEIE (Habilita la interrupción por fin de escritura en la EEPROM de datos) Dirección INTCON,6. Cuando este bit está a "1" la interrupción cuando acaba la escritura en la EEPROM de datos es posible.

GIE (Habilita las interrupciones globalmente) Dirección INTCON,7. Este bit permite que cualquier interrupción de las anteriores sea posible. Para usar alguna de las interrupciones anteriores hay que habilitarlas globalmente e individualmente.

* Estos conceptos no serán explicados aquí ya que no los considero importantes para introducirles en la programación de PICs.

Ahora vamos con el banco 1, solo un comentario antes de empezar: ¿recuerdas la tabla de registros internos que veíamos en el punto 2.2? ves que los registros del banco 0 y los del banco 1 tienen direcciones distintas, en realidad podemos utilizar las mismas direcciones para referirnos a registros que están en uno u otro banco. El pic las diferenciará sin problemas gracias al bit de selección de banco (RP0). Por ejemplo, la dirección 05h se refiere a PORTA si estamos en el banco 0 y a TRISA si estamos en el banco 2.

Sabiendo esto vamos con los registros del BANCO 1:

INDF, Dirección 00h ó 80h, EL registro INDF no es un registro físico. Direccionando INDF actualmente registro el registro quien direcciones están contenidas dentro del FSR registro (FSR es un puntero). Este es una dirección indirecta.

OPTION, Dirección 01h, (configuración del prescaler, Timer, y alguna cosa más) Se estudia bit a bit.

PS0, PS1 y PS2 (Bits del 0 al 2) Configuración del prescaler: El prescaler es un divisor de pulsos que está a la entrada del Timer-contador. El prescaler divide el número de pulsos que le entran al timer-contador o al Wachtdog. El factor de división es el siguiente (según los valores de PS2, PS1 y PS0 respectivamente.

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

6.3. Imagen donde se muestran las terminales del microcontrolador PIC16F452

PSA, Dirección OPTION, 3. Bit de asignación de prescaler: si está a "1" el prescaler se asigna a WDT (Wachtdog), si está a "0" se asigna al TMR0.

TOSE, Dirección OPTION, 4. Bit de selección del tipo de flanco para TMR0. A "1" se incrementa TMR0 por flanco descendente de RA4, a "0" se incrementa TMR0 por flanco ascendente de RA4.

TOCS, Dirección OPTION, 5. Selecciona la entrada de reloj de TMR0. A "1" la entrada de reloj de TMR0 es por flanco de la patilla RA4, a "0" la entrada de reloj de TMR0 es por ciclo de reloj interno.

INTEDG, Dirección OPTION, 6. Tipo de flanco para la interrupción por RB0: A "1" la interrupción será por flanco ascendente, a "0" la interrupción será por flanco descendiente.

RBPU, dirección OPTION, 7. Carga Pull-Up en puerto B. A "0" todas las salidas del puerto B tendrán una carga de pull-Up interna.

PCL, Dirección 02h, igual que en el banco 0

STATUS, Dirección 03h o 83. El registro STATUS contiene el estado aritmético de la ALU. El reset y el bit de selección del Bank 0 o Bank 1 para la memoria de datos.

Como con cualquier registro, El registro STATUS puede ser el destino para alguna instrucción. Si el registro STATUS es el destino para alguna instrucción que afecte la Z, DC o C bits, entonces la escritura de aquellos tres bits esta deshabilitada. Aquellos bits son Activos o Desactivados concordando con la lógica del dispositivo. Además, el TO y PD bits no son escribibles. Por lo tanto, el resultado de una instrucción con el registro STATUS como un destino puede ser diferente que intentado.

Por ejemplo, CLRF STATUS limpiara los tres bits superiores y configuramos el Bit de Z. Esto deja el registro STATUS como 000u u1uu (donde u = sin cambio).

Solo las instrucciones BCF, BSF, SWAPF y MOVWF pueden ser usadas para alterar el registro STATUS (TABLA del SET de Instrucciones), porque aquellas instrucciones no afectan algún bit del STATUS.

BIT 0: C: ACARREO/TOMA negado bit (ADDWF, ADDLW, SUBLW, SUBWF instrucciones) (para el TOMA negado, la polaridad esta invertido)

1 = Un acarreo – fuera desde el bit mas significativo del resultado ocurrido.

0 = No acarreo – fuera desde el bit mas significativo del el resultado ocurrido.

NOTA: Una sustracción fue ejecutado por sumar los dos complementos de los registros de los segundos operandos. Para las instrucciones rodar (RRF, RLF) este bit es leído con el mayor de los bits de mayor o menor orden de el registro fuente.

El BIT 1. DC: bit de acarreo de numero NOTA:

R = bit que puede ser leído.
W = bit que puede ser escrito.
U = bit sin implementar. Lectura como ‘0’.
-n = valor en POR.
‘1’ = Bit esta encendido.
‘0’ = BIT esta Limpio.
x = El bit es desconocido
FSR, Dirección 04h, Igual que en el banco 0

TRISA, Dirección 05h, Configura el puerto A como entrada o salida. Si un bit de este registro se pone a "0" el pin correspondiente en el puerto A será una salida, por el contrario, si se pone a "1" el pin correspondiente en el puerto A será una entrada.

TRISB, Dirección 06h, Igual que el anterior pero con el puerto B

Dirección 07h, No usada en este pic.

EECON1, Dirección 08h, Controla la lectura y escritura en la EEPROM de datos. Se estudia bit a bit.

RD, Dirección EECON1, 0 (bit 0) A "1" iniciamos el ciclo de lectura, cuando acaba el ciclo se pone a "0" el solito.

WR, Dirección EECON1, 1 (bit 1) A "1" indicamos que comienza el ciclo de escritura, cuando acaba el ciclo se pone a "0" él solito.

WREN, Dirección EECON1, 2 (bit 2) si lo ponemos a "1" se permite la escritura, a "0" no se permite.

WRERR, Dirección EECON1, 3 (bit 3) error de escritura, si está a "1" indica que no se ha terminado el ciclo de escritura.

EEIF, Dirección EECON1, 4 (bit 4) interrupción de ciclo de escritura de la EEPROM, si está a "1" indica que el ciclo de escritura ha terminado, hay que ponerlo a "0" por programa.

Bits del 5 al 7 no se utilizan.

EECON2, Dirección 09h, Se utiliza para la escritura en la EEPROM de datos como medida de seguridad: para poder guardar algo en la EEPROM hay que cargar el valor 55h en este registro.
PCLATH, Dirección 0Ah, Igual que en el banco 0.

REGISTRO INTCON

El registro INT CON es un registro que se puede leer o escribir que contiene varios de los bits que permiten las interrupciones para todas las fuentes.

INTCON, Dirección 0Bh, REGISTRO 2-3 INTCON REGISTER (Dirección 0Bh, 8Bh).

BIT 0 RBIF: RB bit de bandera de interrupción de cambio de bit.

1 = En uno de los últimos pines RB7:RB4 ha cambiado el estado.

0 = Ninguno de los pines RB7:RB4 ha cambiado de estado

BIT 1 INTF: RB0 / INT bit de bandera de interrupción externa.

1 = El RB0/INT una interrupción externa a ocurrido

0 = El RB0/INT ninguna interrupción externa a ocurrido

BIT 2 T0IF: TMR0 bit de bandera de interrupción de sobréflujo.

1 = El registro TMR0 se ha sobrecargado

0 = El registro TMR0 no se ha sobrecargado

BIT 3 RBIE: RB Port bit que permite la interrupción cambios del puerto.

1 = Permite la interrupción de cambio del puerto RB

0 = Deshabilita la interrupción de cambio del puerto RB

BIT 4 INTE: RB0/INT Bit que permite la Interrupción Externa

1 = Permite la interrupción RB0 / INT externa

0 = Deshabilita la interrupción RB0 / INT externa
BIT 5 T01 : TMR0 bit que permite la interrupción de sobrecarga

1 = Habilita la interrupción TMR0

0 = Deshabilita la interrupción TMR0

BIT 6 EEIE: bit que permite las interrupciones completen escritura en EE:

1 = Habilita las interrupciones de escritura por completo en EE

0 = Deshabilita las interrupciones de escritura por completo EE

BIT 7 GIE: bit que permite las interrupciones Globales.

1 = Permite todas las interrupciones

0 = Deshabilita todas las interrupciones

Los bits de bandera de las interrupciones son colocadas en alto cuando ocurre una condición de la interrupción.

Set de Instrucciones del PIC16F84

Para entender mejor cada instrucción se explica a continuación el significado de algunos parámetros:

F: Registro al que afecta la instrucción. Registro de dirección de fila (0x00 a 0x7F)

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.

Instrucciones orientadas a registros.

ADDWF f,d - Suma el valor del registro F con el valor de W y deja el resultado donde indique D. Suma W y el registro f, el resultado lo guarda según d (si d=0 se guarda en W y si d=1 se guarda en f).

ANDWF f,d - Realiza la operación AND lógica entre W y f, el resultado lo guarda según d.

CLRF f - Borra el registro f (pone todos sus bits a cero).

CLRW - Borra el acumulador

COMF f,d Calcula el complementario del registro f (los bits que están a "0" los pone a "1" y viceversa. Resultado según d.

DECF f,d Decrementa a f en uno (le resta uno). Resultado según d.

DECFSZ f,d Decrementa f y se salta la siguiente instrucción si el resultado es cero. Resultado según d.

INCF f,d Incrementa f en uno (le suma uno). Resultado según d.

INCFSZ f,d Incrementa f y se salta la siguiente instrucción si el resultado es cero (cuando se desborda un registro vuelve al valor 00h). Resultado según d.

IORWF f,d Suma lógica del valor del registro F con el valor del registro W. El resultado se guarda donde indique D. Realiza la operación lógica OR entre W y f. Resultado según d.

MOVF f,d Mueve el contenido del registro f a W si d=0 (si d=1 lo vuelve a poner en el mismo registro)

MOVWF f mueve el valor de W a f. Por ejemplo, si queremos copiar el valor del registro "REG1" al registro "REG2" (ya veremos como ponerles nombres a los registros) escribiremos:

MOVF REG1,0; mueve el valor de REG1 a W. Movimiento del valor del registro F a W (D=0) o a si mismo (D=1)

MOVWF REG2; mueve el valor de W a REG2 Lo que va después del; son comentarios.

NOP - No hace nada, solo pierde el tiempo durante un ciclo.

RLF f,d Rota el registro f hacia la izquierda a través del bit CARRY (todos los bits se mueven un lugar hacia la izquierda, el bit 7 de f pasa al CARRY y el bit CARRY pasa al bit 0 de f). Resultado según d.

RRF f,d Lo mismo que RLF pero hacia la derecha.

SUBWF f,d Resta f y W (f - W). Resultado según d.

SWAPF f,d intercambia los 4 primeros bit de f por los otros cuatro. Resultado según d.

XORWF f,d Realiza la operación lógica XOR (OR exclusiva) entre W y f. Resultado según d.

Instrucciones orientadas a bits:

BCF f,b Pone a "0" el bit b del registro f.

BSF f,b Pone a "1" el bit b del registro f.

BTFSC f,b Se salta la siguiente instrucción si el bit b del registro f es "0"

BTFSS f,b. Bit Test f, Skip if Set. Se salta la siguiente instrucción si el bit b del registro f es "1". Si es “0” continua la instrucción inmediata siguiente.

Instrucciones orientadas a constantes y de control:

ADDLW k Le suma el valor k al acumulador (W).

ANDLW k Operación lógica AND entre W y el valor k (resultado en W).

CALL k Llamada a subrutina cuyo inicio esta en la dirección k

CLRWDT - Borra el registro Watchdog.

GOTO k Salta a la dirección k de programa.

IORLW k Suma lógica del valor literal k con el valor del registro W. el resultado es guardado en el registro W. Operación lógica OR entre W y el valor k (resultado en W).

MOVLW k Movimiento del valor literal k al registro de trabajo w. carga el acumulador con el valor k. Por ejemplo, si queremos cargar el valor 2Ah en el registro "REG1" escribiremos:

MOVLW 2AH carga el acumulador W con el valor 2Ah

MOVWF REG1 Movimiento del valor del registro de trabajo w al registro F. mueve el valor de W a "REG1".

RETFIE - Instrucción para volver de la interrupción

RETLW k carga el valor k en W y vuelve de la interrupción

RETURN - vuelve de una subrutina.

SLEEP - El pic pasa a modo de Standby.

Instrucciones para el lenguaje ensamblador

Podemos usar para escribir los programas el bloc de notas de Windows, una vez escrito se guarda con extensión .asm y se convierte (ensambla) con un programa ensamblador, el MPASM. El resultado es un archivo con extensión .hex que podemos transferir al PIC16F84. Todo esto se explica más detalladamente en Programador del PIC16F84.

Existen una serie de instrucciones que son para el ensamblador y nos hacen la tarea de programación más sencilla y más legible.

EQU: Un ejemplo de esto son las etiquetas, podemos poner un nombre a un registro de memoria, esto se hace mediante la instrucción EQU. Por ejemplo:

```
VARIABLE1    EQU    0CH
```

A partir de ahora en lugar de escribir 0CH podemos escribir VARIABLE1

Con EQU también podemos poner nombre a constantes de la misma forma.

#DEFINE: Otra instrucción para el ensamblador que usaremos será la instrucción #DEFINE. Es parecido a EQU, solo que aquí no ponemos etiquetas a un registro, podemos ponerla a una instrucción entera, Por ejemplo:

```
#DEFINE    BANCO1    BSF   STATUS, 5
#DEFINE    BANCO0    BCF   STATUS, 5
```

A partir de ahora, cuando escribamos BANCO1 se pondrá a "1" el bit de selección de banco y pasaremos al banco 1, al escribir BANCO0 pasaremos al banco 0.
ORG: Indica al ensamblador la dirección (de memoria de programa) donde se guardará la instrucción que vaya a continuación. Por ejemplo:

```
ORG    00H
CLRF   VARIABLE1
```

La instrucción CLRF está en la dirección de memoria de programa 00H (será la primera instrucción en ser ejecutada por el pic)

END: Se escribe al final del programa para indicar que ya ha acabado. (es obligatorio, si no da error).

Etiquetas a direcciones de Programa: muy útiles para usar con instrucciones CALL (Llamada a subrutina) o GOTO (Salto). Por ejemplo:

```
.........
[Hay programa anterior]
.........
BTFSC   VARIABLE1, 0        ; Si el bit 0  de VARIABLE1 es
                            ;"0" se salta la siguiente
                            ; Instrucción
GOTO    ESUNO               ; Salta a ESUNO solo si el bit 0
```

```
                                ; De VARIABLE1 es "1"
BSF     VARIABLE1, 0             ; Si el bit 0 de VARIABLE1 es 0
                                ; Se ejecuta esta instrucción y el
                                ; Programa sigue por aquí
.............
[Continúa el programa]
.............
ESUNO                           ; Etiqueta a una dirección de
                                ; programa
BCF     VARIABLE1, 0             ; Si el bit 0 de VARIABLE1 es
                                ;"1" se ejecuta esta otra
                                ; Instrucción y el programa
                                ; sigue por aquí
......
[Continúa el programa]
......
```

Un poco de orden:

Es importante llevar un poco de orden a la hora de escribir el programa, nos ayudará mucho:

Al principio van los EQU y los #DEFINE, después comenzamos con el programa.

El programa se escribe en cuatro columnas separadas por tabuladores:

En la primera columna se ponen las etiquetas a direcciones de programa

En la segunda columna se ponen las instrucciones (BSF, CLRF, BTFSC... etc.)

En la tercera columna se ponen Los registros o parámetros a los que afecta la instrucción.

En la cuarta columna se ponen los comentarios que creas pertinentes (cuantos mas mejor) seguidos de un punto y coma.

Nótese que yo he echo un tipo de caja usando los puntos y comas, esto solo tómalo como un ejemplo.

Primer ejemplo:

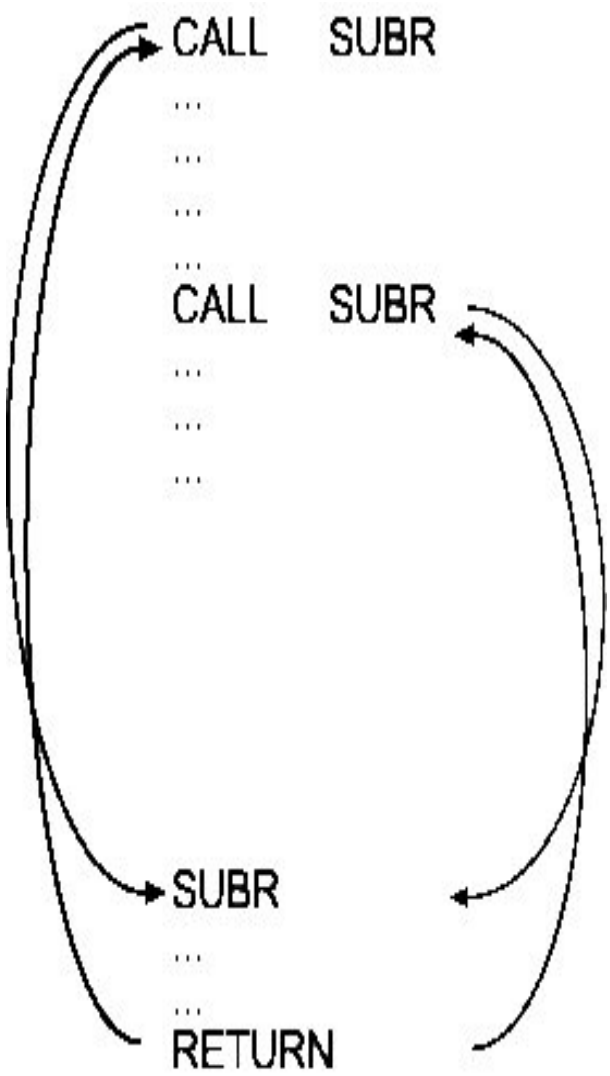
```
;* EJEMPLO 1: PROGRAMA BIEN ORDENADO*
;**********************************
;* El siguiente programa configura  *
;* RA1 como entrada y RA0 como      *
;* salida y hace que la salida (RA0)*
;* sea la inversa de la entrada     *
;* (RA1)                            *
;**********************************

;(Primero los ECU y los #DEFINE)

STATUS  EQU 03H
TRISA   EQU 05H
PORTA   EQU 05H

#DEFINE BANC00  BCF STATUS,5
#DEFINE BANC01  BSF STATUS,5

;(Después empezamos con el programa)

    ORG 00H             ;Empezamos siempre a escribir en esta dirección
    BANC01              ;Pasamos al banco 1 para hacer algunas
                        ;configuraciones
    BCF TRISA,0         ;Configuramos RA0 como salida
    BSF TRISA,1         ;Configuramos RA1 como entrada
    BANC00              ;Volvemos al banco 0

INICIO  BTFSC   PORTA,1 ;Comprueba la entrada (RA1), si es "0" se salta la
                        ;siguiente instrucción
        GOTO    ESUNO   ;si la entrada (RA1) es "1" va a ESUNO

        BSF     PORTA,0 ;Pone a "1" la salida RA0. Ejecuta esta instrucción
                        ;porque la entrada RA1 era "0"
        GOTO    INICIO  ;Vuelve otra vez a comprobar el estado de la
                        ;entrada RA1

ESUNO   BCF     PORTA,0 ;Pone a "0" la salida RA0. Ejecuta esta instrucción
                        ;porque la entrada RA1 era "1"
        GOTO    INICIO  ;Vuelve otra vez a comprobar el estado de la
                        ;entrada RA1

    END
```

Subrutinas

Una subrutina o subprograma es una parte de programa que hace algo concreto y se repite a menudo, para ahorrar memoria y esfuerzo y para hacer el programa mas comprensible se agrupa en forma de subrutina. Una subrutina se debe ejecutar siempre llamándola con la instrucción CALL y al final de dicha subrutina debe haber siempre un RETURN. El esquema de la derecha muestra como funcionan las subrutinas:

Durante el programa principal se llama varias veces a la subrutina SUBR (el nombre es lo de menos) con la instrucción CALL. Cuando el pic ejecuta una instrucción CALL se guarda en memoria la dirección de código de programa a la que tiene que retornar de tal forma que cuando se encuentra con la instrucción RETURN vuelve al programa principal donde lo dejó.

Una subrutina no solo puede ser llamada desde el programa principal, también puede hacerse desde otra subrutina (una subrutina que llama a otra subrutina) o desde una interrupción (enseguida las veremos).



6.4. Imagen donde se ejemplifica el ciclo de una subrutina.

El siguiente ejemplo muestra un programa que utiliza una subrutina de retardo a la que llama DELAY. Esta subrutina de retardo se hace decrementando el registro CUENTA2 desde FFh hasta 00h 16 veces (las veces que se decrementa CUENTA2 son contadas hacia atrás por CUENTA1) De esta forma se consigue perder tiempo (el tiempo perdido con esta subrutina depende de la frecuencia a la que opere el pic).

Segundo ejemplo

```
;*      EJEMPLO 2 USO DE SUBRUTINAS       *
;****************************************
;* Este programa configura RB0 como salida*
;* y genera una intermitencia en dicha    *
;* salida                                 *
;****************************************

STATUS  EQU 03H
TRISB   EQU 06H
PORTB   EQU 06H

CUENTA1 EQU 0CH     ;Las variables que usemos siempre a
                    ;partir de la dirección 0Ch
CUENTA2 EQU 0DH

F   EQU 1
w   EQU 0


    ORG 00H
    BSF STATUS,5            ;banco 1
    BCF TRISB,0             ;RB0 como salida
    BCF STATUS,5            ;banco 0

INICIO  BSF     TRISB,0     ;Pone a "1" RB0 (enciende)
        CALL    DELAY       ;Llama a la subrutina de retardo
        BCF     TRISB,0     ;Cuando vuelve del retardo pone
                            ;a "0" RB0 (apaga)
        CALL    DELAY       ;llama a la subrutina de retardo
        GOTO    INICIO      ;cuando vuelve del retardo


;=================================================
;=  DELAY:  Subrutina de retardo              =
;=      Modifica los siguientes registros: =
;=          CUENTA1                    =
;=          CUENTA2                    =
;=          ACUMULADOR                 =
;=          STATUS                     =

;(Conviene hacerse un pequeño resumen de lo que
;hace cada subrutina, puede sernos muy útil para
;usarla en otros programas)

DELAY   MOVLW   010H        ;Carga el acumulador con el valor
                            ;10H (16 en decimal)
        MOVWF   CUENTA1     ;Mueve el contenido del acumulador
                            ;a CUENTA1
ACA1    MOVLW   0FFH        ;Carga el acumulador con el valor FFH
ACA     DECFSZ  CUENTA2,F   ;Decrementa CUENTA2, guarda el resultado
                            ;en f, y si es cero se salta la siguiente
                            ;instrucción
        GOTO    ACA         ;vuelve a decrementar mientras
                            ;CUENTA2 no sea cero
        DECFSZ  CUENTA1,F   ;Se decrementa CUENTA1 cada vez que
                            ;CUENTA2 llega a cero
        GOTO    ACA1        ;mientras CUENTA1 no llegue a cero recarga
                            ;CUENTA2 y repite el proceso
        RETURN              ;retorna al programa principal

;=                                              =
;=  FIN DE LA SUBRUTINA DELAY                   =
;=================================================

        END
```

Interrupciones

Cuando se produce una interrupción el pic deja automáticamente lo que esté haciendo, va directo a la dirección 04h de programa y ejecuta lo que encuentre a partir de allí hasta encontrarse con la instrucción RETFIE que le hará abandonar la interrupción y volver al lugar donde se encontraba antes de producirse dicha interrupción.
Para que se pueda producir una interrupción hay que habilitar las interrupciones globalmente y la interrupción en concreto que queremos utilizar (con el registro **INTCON**). Este pic tiene 4 tipos de posibles interrupciones:
Por cambio en los bits RB4-RB7
Por el estado de RB0
Por desbordamiento del Timer-contador
Por fin de ciclo de escritura de la EEPROM de datos
Mientras se está ejecutando una interrupción no se puede producir otra interrupción, el pic no lo permite.
Una cosa importante a tener en cuenta al usar interrupciones es que cuando estas se producen podríamos estar trabajando con registros que pueden ser modificados en la propia interrupción, como el acumulador o el **STATUS**. Para que la interrupción no eche a perder el buen funcionamiento del programa principal conviene guardar los valores de estos registros en otras variables que no vayamos a modificar. Antes de salir de la interrupción volvemos a restaurar los valores guardados y todo solucionado.
El siguiente ejemplo muestra un programa que usa la interrupción por cambio en el puerto B (En pines RB4 a RB7).

Fíjate que cuando se acciona el pulsador la entrada RB0 se pone a "0". Para evitar contar los rebotes se llama a una subrutina de retardo llamada REBOTE, esta subrutina funciona bien para osciladores de 4MHz.

Ejemplo: 3

```
;*****************************************
;*    EJEMPLO 3: USO DE INTERRUPCIONES     *
;*****************************************
;* Este programa invierte el estado del pin*
;* RA0 cada vez que se modifica el estado  *
;* de alguno de los pines RB4, RB5, RB6 o  *
;* RB7. Para ello habilita la interrupción *
;* por cambio de RB4-RB7                   *
;*****************************************

STATUS  EQU 03H
TRISA   EQU 05H
PORTA   EQU 05H
TRISB   EQU 06H
PORTB   EQU 06H
INTCON  EQU 0BH

ACUM    EQU 0CH
STAT    EQU 0DH

F   EQU 1
w   EQU 0

#DEFINE BANCO0  BCF STATUS,5
#DEFINE BANCO1  BSF STATUS,5

        ORG 00H
        GOTO    INICIO  ;ponemos este GOTO al principio para poder poner
                        ;el subprograma de las interrupciones a partir de
                        ;la dirección 04h
                        ;Comienza la interrupción:
                        ;=========================

        ORG 04H         ;El pic salta a esta dirección cuando se produce
                        ;una interrupción
        BCF INTCON,0    ;bit que indica un cambio en RB4-RB7, recuerda que
                        ;hay que ponerlo a "0" por programa, este es el
                        ;momento
                        ;comenzamos guardando el contenido del acumulador
                        ;y del STATUS para restaurarlos antes de salir de
                        ;la interrupción (es recomendable hacer esto
                        ;siempre que se usen interrupciones)

        MOVWF   ACUM    ;Copia el acumulador al registro ACUM
        MOVF    STATUS,W ;Guarda STATUS en el acumulador
        BANCO0          ;Por si acaso, nunca se sabe en que parte de
                        ;programa principal salta la interrupción
        MOVWF   STAT        ;Copia el acumulador al registro STAT
                        ;Invertimos el estado de RA0:
                        ;============================
        BTFSC   PORTA,0 ;si RA0 es "0" salta la siguiente instrucción
        GOTO    ESUNO   ;vete a ESUNO
        BSF     PORTA,0 ;Pon a "1" RA0 (porque era "0")
        GOTO    HECHO   ;ya está invertido RA0, vete a HECHO
```

```
ESUNO   BCF PORTA,0       ;Pon a "0" RA0 (Porque era "1")

                          ;Ya se ha invertido el estado de RA0
                          ;==================================
                          ;ahora hay que restaurar los valores del STATUS y
                          ;del acumulador antes de salir de la interrupción:

HECHO   MOVF    STAT,W    ;Guarda el contenido de STAT en el acumulador
        MOVWF   STATUS    ;Restaura el STATUS
        SWAPF   ACUM,F    ;Da la vuelta al registro ACUM
        SWAPF   ACUM,W    ;Vuelve a dar la vuelta al registro ACUM y restaura
                          ;el acumulador (Con la instruccion SWAPF para no
                          ;alterar el STATUS, la instrucción MOVF altera el
                          ;bit 2 del STATUS)
        RETFIE            ;fin de la interrupción
                          ;Fin de la interrupción
                          ;======================

INICIO  BANCO1            ;Pasamos al banco 1
        MOVLW   0FFH      ;Todos los bits del acumulador a "1"
        MOVWF   TRISB     ;configuramos todo el puerto B como entradas
        BCF     TRISA,0   ;RA0 como salida
        BANCO0            ;Volvemos al banco 0

                          ;Configuración de las interrupciones:
                          ;====================================

        BSF INTCON,7      ;Habilita las interrupciones globalmente
        BSF INTCON,3      ;Habilita la interrupción por cambio de puerto B

                          ;====================================
                          ;ya están configuradas las interrupciones, a
                          ;partir de ahora cuando haya un cambio en RB4-RB7
                          ;saltará la interrupción (a la dirección 04h de
                          ;programa)

NADA    GOTO    NADA      ;En este ejemplo no se hace nada en el programa
                          ;principal, simplemente se espera a que salte la
                          ;interrupción. La verdadera utilidad de las
                          ;interrupciones es que se pueden hacer "cosas"
                          ;mientras sin preocuparse de la interrupción

        END               ;FIN DE PROGRAMA
```

Timer - Contador TMR0

El registro TMR0 puede contar ciclos de instrucción interna o pulsos de entrada por RA4 según el valor del bit 5 del registro OPTION (TOCS). Si este bit está a "1" TMR0 cuenta pulsos por RA4 y se le llama Contador; si el bit está a "0" cuenta ciclos de instrucción interna y se le llama Timer.

Cada ciclo de instrucción dura 4 veces el ciclo del reloj del pic (para un reloj de 4MHz ==> Ciclo reloj=0,25 µSeg ==> Ciclo instrucción = 4 X 0,25 = 1µSeg).

Cuando lo usamos como contador (Por RA4) podemos determinar si el incremento se hará por flanco ascendente o descendente con el bit 4 del registro OPTION (TOSE).

Podemos leer o escribir el registro TMR0 en cualquier momento. Cuando escribamos en él deja de contar durante dos ciclos, cuando lo leamos no pasa nada.

Podemos asignar el prescaler al TMR0, si hacemos esto podemos elegir el factor en el que se verá dividido el conteo mediante los bits del 0 al 2 del registro OPTION según la tabla del factor de división. Por ejemplo, si elegimos un factor de división de 1/2 tienen que entrar 2 pulsos para que TMR0 se incremente en uno, si está a 1/4 tienen que entrar 4... etc.

También podemos utilizar la interrupción que se produce cuando se desborda el TMR0, es decir, cuando pasa de FFh a 00h. (Se configura desde el registro INTCON).

El siguiente ejemplo usa la interrupción por desbordamiento de TMR0 para obtener una onda cuadrada a la salida RB0.

Ejemplo 4

```
;*         EJEMPLO 4: USO DEL TMR0         *
;*****************************************
;* Este programa crea una señal cuadrada a *
;* la salida RB0, para ello utiliza el TMR0*
;* y la interrupción por desbordamiento del*
;* mismo. Se le asignará el prescaler con  *
;* un factor de división de 1/2. De esta   *
;* forma las interrupciones saltarán a     *
;* intervalos fijos de tiempo. Invirtiendo *
;* el estado de RB0 durante las            *
;* interrupciones se conseguirá una onda   *
;* cuadrada perfecta                       *
;*****************************************

OPTIONR EQU 01H ;Registro para configuración del TMR0
STATUS  EQU 03H
TRISB   EQU 06H
PORTB   EQU 06H
INTCON  EQU 0BH

ACUM    EQU 0CH
STAT    EQU 0DH

F   EQU 1
w   EQU 0

#DEFINE BANC00  BCF STATUS,5
#DEFINE BANC01  BSF STATUS,5

        ORG 00H
        GOTO    INICIO      ;ponemos este GOTO al principio para poder poner
                            ;el subprograma de las interrupciones a partir de
                            ;la dirección 04h

                            ;Comienza la interrupción:
                            ;=========================

        ORG 04H             ;El pic salta a esta dirección cuando se produce
                            ;una interrupción
        BCF     INTCON,2    ;bit que indica desbordamiento de TMR0, recuerda
                            ;que hay que ponerlo a "0" por programa, este es
                            ;el momento

                            ;comenzamos guardando el contenido del acumulador
                            ;y del STATUS para restaurarlos antes de salir de
                            ;la interrupción (es recomendable hacer esto
                            ;siempre que se usen interrupciones)

        MOVWF   ACUM        ;Copia el acumulador al registro ACUM
        MOVF    STATUS,W    ;Guarda STATUS en el acumulador
        BANC00              ;Por si acaso, nunca se sabe en que parte de
                            ;programa principal salta la interrupción
        MOVWF   STAT        ;Copia el acumulador al registro STAT

                            ;Para generar una onda cuadrada Invertimos el
                            ;estado de RB0 cada vez que salta una interrupción
                            ;================================================

        BTFSC   PORTB,0     ;si RB0 es "0" salta la siguiente instrucción
        GOTO    ESUNO       ;vete a ESUNO
        BSF     PORTB,0     ;Pon a "1" RB0 (porque era "0")
        GOTO    HECHO       ;ya está invertido RB0, vete a HECHO
```

```
ESUNO   BCF PORTB,0         ;Pon a "0" RA0 (Porque era "1")

                            ;Ya se ha invertido el estado de RB0
                            ;===================================

                            ;ahora hay que restaurar los valores del STATUS y
                            ;del acumulador antes de salir de la interrupción:

HECHO   MOVF    STAT,W      ;Guarda el contenido de STAT en el acumulador
        MOVWF   STATUS      ;Restaura el STATUS
        SWAPF   ACUM,F      ;Da la vuelta al registro ACUM
        SWAPF   ACUM,W      ;Vuelve a dar la vuelta al registro ACUM y restaura
                            ;el acumulador (Con la instruccion SWAPF para no
                            ;alterar el STATUS, la instrucción MOVF altera el
                            ;bit 2 del STATUS)
        RETFIE              ;fin de la interrupción

                            ;Fin de la interrupción
                            ;======================

INICIO  BANCO1              ;Pasamos al banco 1
        BCF TRISB,0         ;RB0 como salida
        BCF OPTIONR,3       ;Asignamos el preescaler a TMR0
        BCF OPTIONR,0       ;\
        BCF OPTIONR,1       ; }Prescaler a 1/2
        BCF OPTIONR,2       ;/
        BCF OPTIONR,5       ;Entrada de TMR0 por ciclo de
                            ;instrucción interna (se pone a contar)
        BANCO0              ;Volvemos al banco 0

                            ;Configuración de las interrupciones:
                            ;====================================

        BSF INTCON,7        ;Habilita las interrupciones globalmente
        BSF INTCON,5        ;Habilita la interrupción por desbordamiento de TMR0
                            ;====================================
                            ;ya están configuradas las interrupciones, a
                            ;partir de ahora cuando cuando se desborde TMR0
                            ;saltará la interrupción (a la dirección 04h de
                            ;programa)

NADA    GOTO    NADA        ;En este ejemplo no se hace nada en el programa
                            ;principal, simplemente se espera a que salte la
                            ;interrupción. La verdadera utilidad de las
                            ;interrupciones es que se pueden hacer "cosas"
                            ;mientras sin preocuparse de las interrupciones

        END                 ;FIN DE PROGRAMA
```

Ejemplos de programas

Ejemplo 1: Programa que configura RA1 como entrada, RA0 como salida y hace que la salida (RA0) sea la inversa de la entrada (RA1).

Ejemplo 2: Uso de subrutinas. Configura RB0 como salida y genera una intermitencia.

Ejemplo 3: Uso de interrupciones. Se invierte el estado del pin RA0 cada vez que se modifica el estado de alguno de los pines RB4, RB5, RB6 o RB7. Para ello habilita la interrupción por cambio de RB4-RB7.

Ejemplo 4: Uso del TMR0. Crea una señal cuadrada a la salida RB0, para ello utiliza el TMR0 y la interrupción por desbordamiento del mismo. Se le asigna el prescaler con un factor de división de 1/2. De esta forma las interrupciones saltan a intervalos fijos de tiempo. Invirtiendo el estado de RB0 durante las interrupciones se consigue una onda cuadrada perfecta.

El temporizador TMR0

Los temporizadores son contadores que se cargan con un valor al comienzo de la cuenta del tiempo y van aumentando con cada impulso de reloj. Cuando se desbordan. O sea, sobrepasan el valor máximo que pueden contener o llegan a cero, colocan un bit señalizador automáticamente a 1, avisando del fin de la cuenta del tiempo. Son muy útiles porque evitan al procesador que tenga que dedicarse en el programa principal a contar tiempos. Están especializados y cuando terminan siempre levanta la bandera; si se desea también pueden provocar una interrupción al procesador.

El TMR0 es un contador de 8 bits ascendente que cuando llega a alcanzar el valor máximo (que es FF Hex) pasa 00 y levanta su bandera colocando a TMROIF = 1, que también se denomina TOIF.

Los impulsos de reloj para el incremento del TMR0 Pueden recibirse desde el exterior a través de la patita RA4/T0CKI, ó bien a una frecuencia de Fosc/4.

Calculo del Tiempo

El tiempo que mide el TMR0 desde que se carga n un valor hasta que se desborda depende del valor del registro TMR0 que ocupa la direccion 1 del banco 0 de la RAM. También depende de la posible actuación del predivisor de frecuencia que puede aplicarse a este temporizador. Este Predivisor divide por un determinado rango los impulsos a aplicar al temporizador. El predivisor puede usarlo el TMR0 o el perro guardian. La fórmula a utilizar para calcular el tiempo es la siguente:

Siendo N el valor en decimal que previamente se carga en TMR0. si por ejemplo se carga un valor de 156 en decimal y el procesador funciona a 4 Mhz recibiendo el oscilador interno los impulsos de reloj, el tiempo que tarda el temporizador hasta desbordarse será:

$$(4)\left(\frac{1}{\text{Cristal}}\right)(256\text{-N})(\text{Predivisor}) = Tseg$$

Ejemplo:

$$(4)\left(\frac{1}{4\ \text{Mhz}}\right)(256\text{-}156)(16) = 1.6\cdot 10^{-3}\,seg$$

$$(4)\left(250\cdot 10^{-9}\right)(100)(16) = 1.6\cdot 10^{-3}\,seg$$

$$(1\cdot 10^{-6})(1600) = 1.6\cdot 10^{-3}\,seg$$

EJEMPLO 2 Para un cristal de 10 Mhz

$$(4)\left(\frac{1}{\text{Cristal}}\right)(256\text{-N})(\text{Predivisor}) = Tseg$$

Ejemplo 2 :

$$(4)\left(\frac{1}{10\ \text{Mhz}}\right)(256\text{-}0)(256) = 26.21x10^{-3}\,seg$$

$$(4)\left(100\cdot 10^{-9}\right)(256)(256) = 26.21x10^{-3}\,seg$$

$$(0.4^{-6})(65536) = 26.21x10^{-3}\,seg$$

Si necesitamos el Tiempo exacto

$$(4)\left(\frac{1}{\text{Cristal}}\right)(256\text{-N})(\text{Predivisor}) = Tseg$$

Ejemplo 3 :

$$(4)\left(\frac{1}{3.2768\text{Mhz}}\right)(256\text{-}192)(256) = 20.0^{-3}\,seg$$

$$(4)\left(305.157155935\cdot 10^{-9}\right)(256\text{-}192)(256) = 20.0x10^{-3}\,seg$$

$$\left(500\cdot 10^{-9}\right)(256-192)(256) = 20.0x10^{-3}\,seg$$

$$\left(500\cdot 10^{-9}\right)(64)(256) = 20.0x10^{-3}\,seg$$

Programación del PIC en MPLAB

Abrimos MPLAB y en el menú de Project seleccionamos Project Wizard.

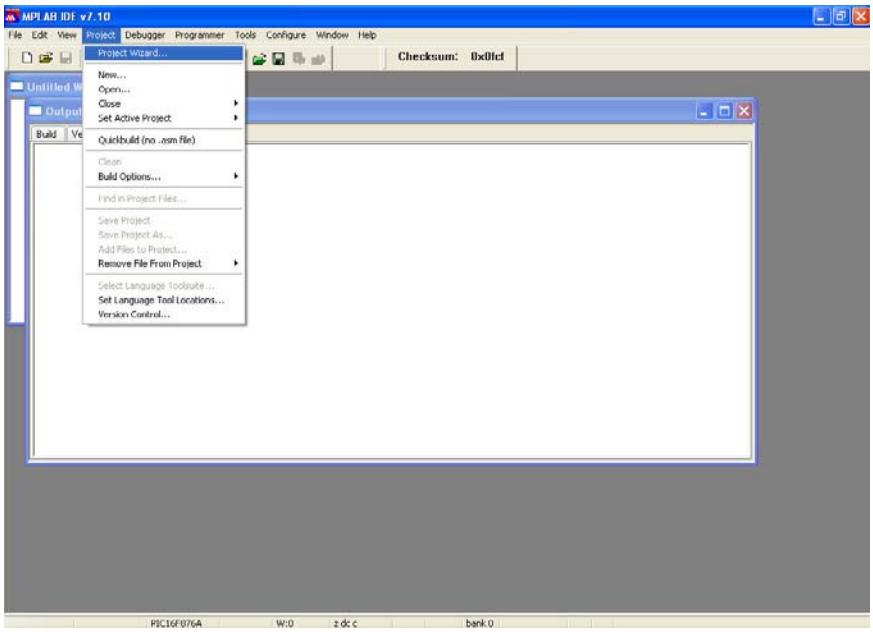


Fig. 6.6. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. En esta imagen se sobresalta el menú Project Wizard.

Una bienvenida y damos siguiente.

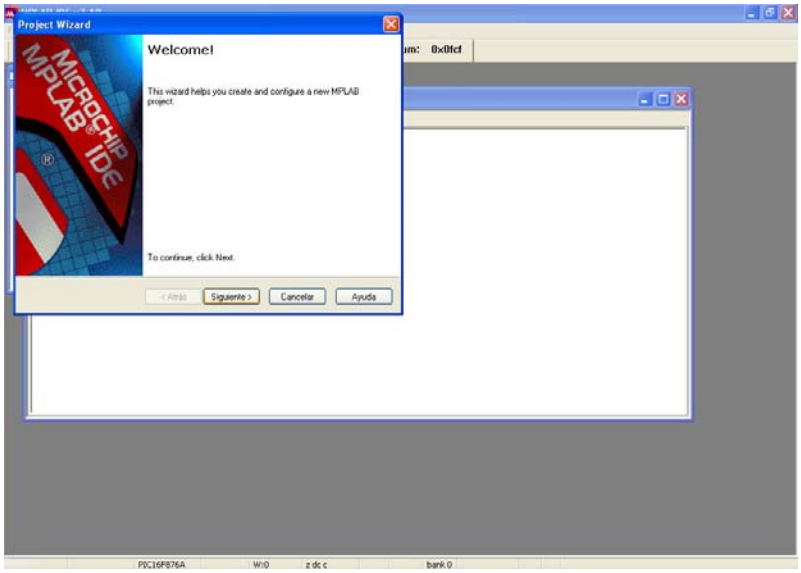


Fig. 6.7. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. En esta imagen se muestra la bienvenida.

Seleccionamos el dispositivo

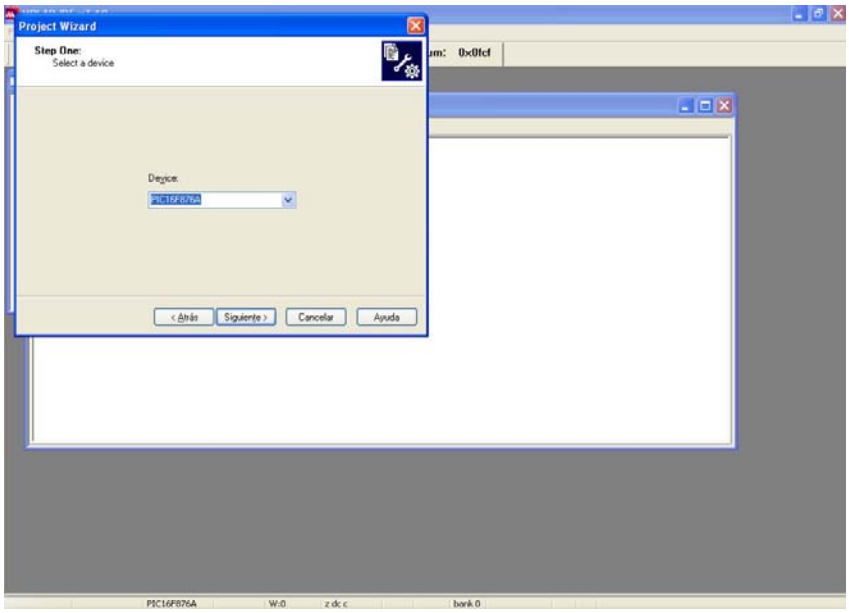


Fig. 6.8. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. En esta imagen se muestra el dispositivo a programar

Las librerías deben estar en sus directorios.

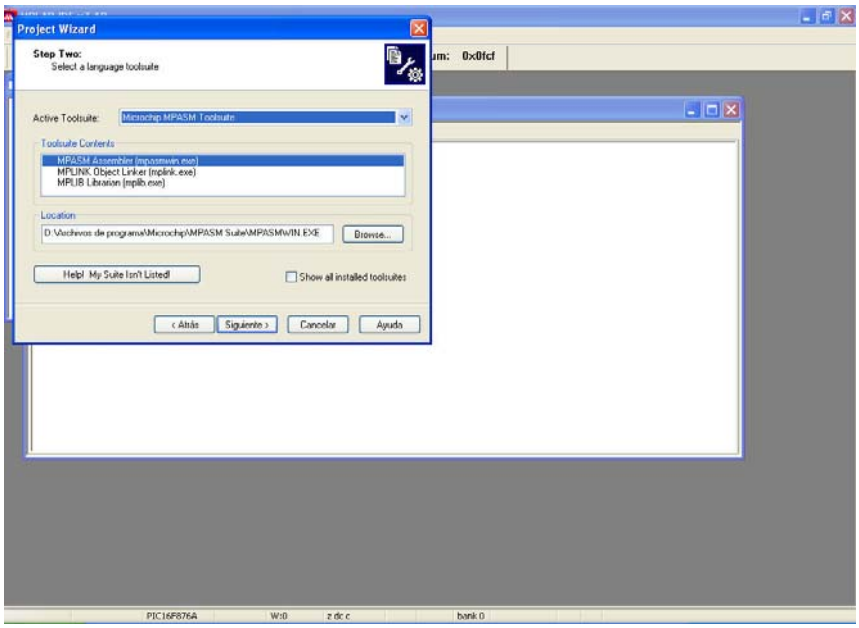


Fig. 6.9. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. En esta imagen se muestra el compilador y archivos que deben estar instalados para su compilación.

Seleccionamos los directorios para el programa a compilar.

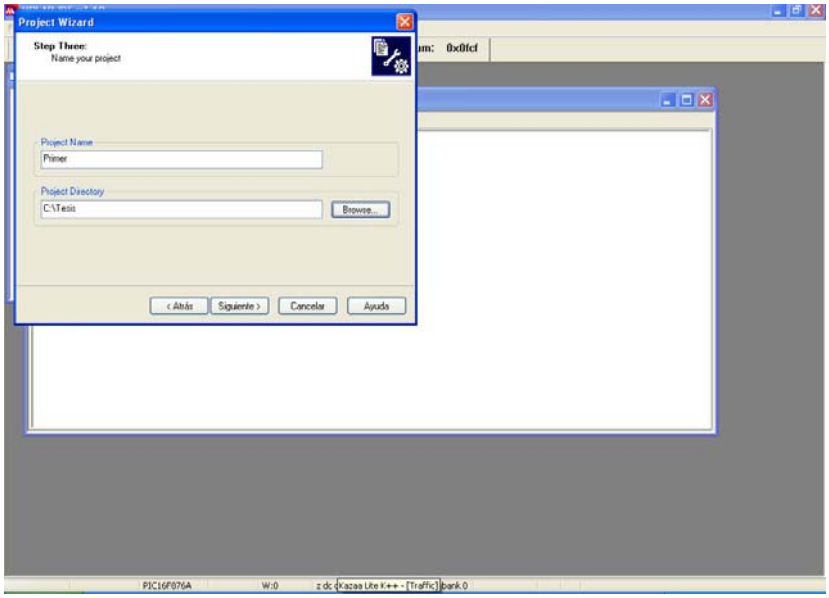


Fig. 6.10. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. En esta imagen se muestra el directorio donde se copiaran los archivos para su compilación.

Seleccionamos el programa previamente hecho.

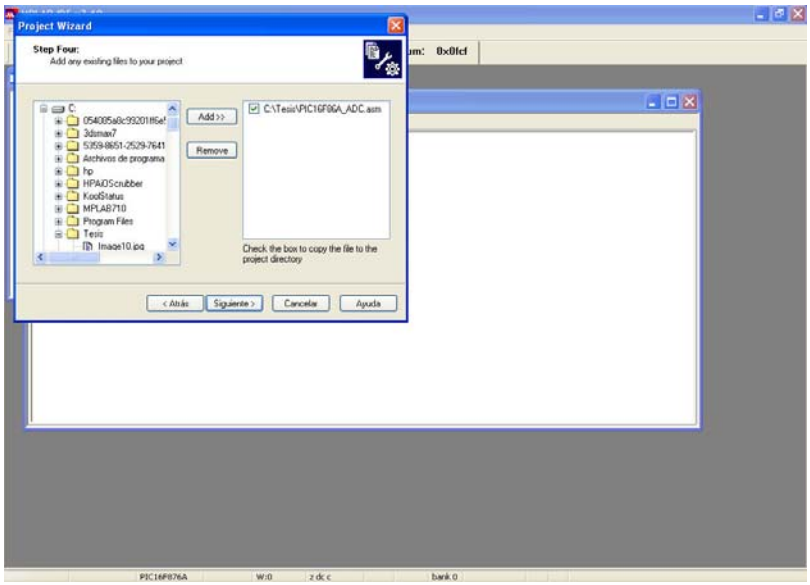


Fig. 6.11. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. En esta imagen se muestra la bienvenida.

Finalizamos el asistente para el proyecto nuevo

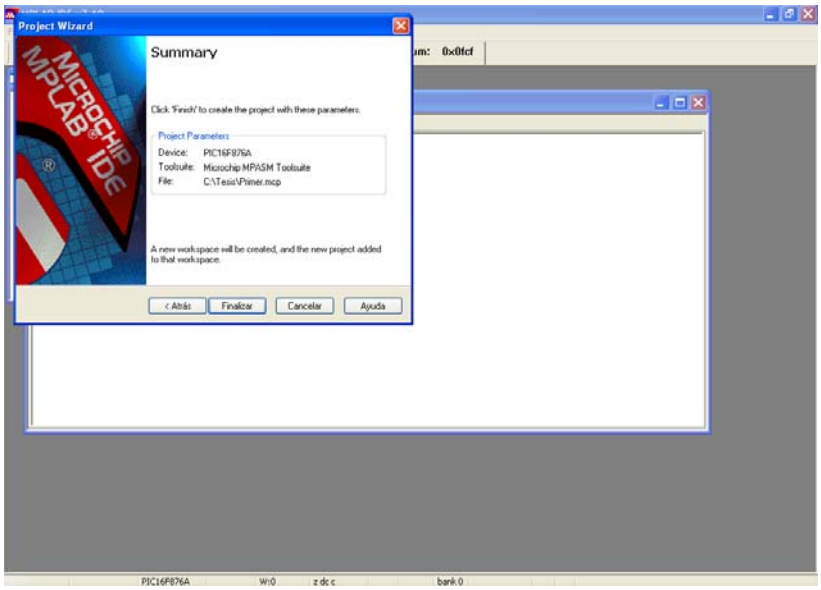


Fig. 6.12. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. En esta imagen se muestra un resumen de lo seleccionado, sí uno se equivoco puede regresar y corregir lo seleccionado.

Inicio de MPLAB con el programa

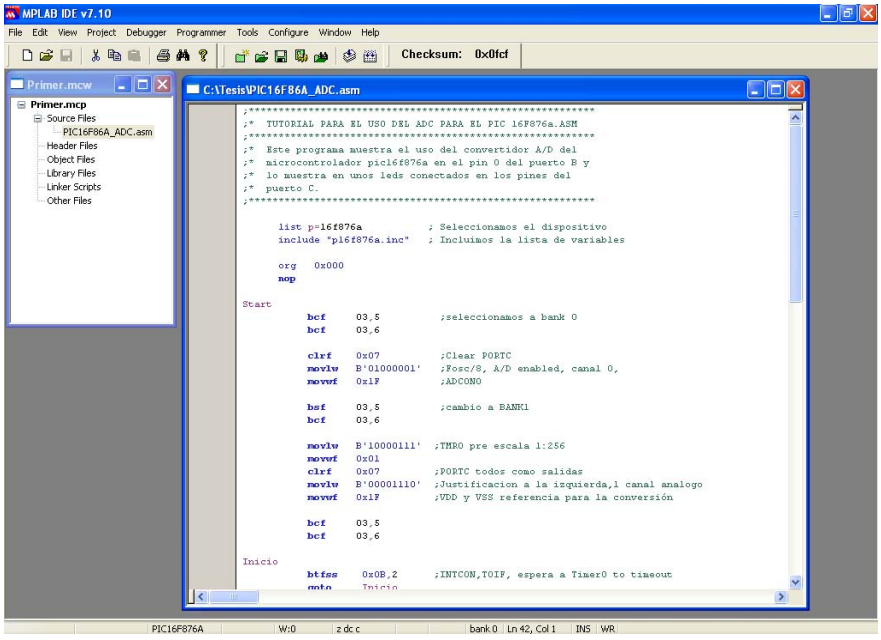


Fig. 6.13. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. En esta imagen se muestra ya el MPLAB con el programa seleccionado.

En el menú de Project, seleccionamos build all y esperamos a que se concluya la compilación.

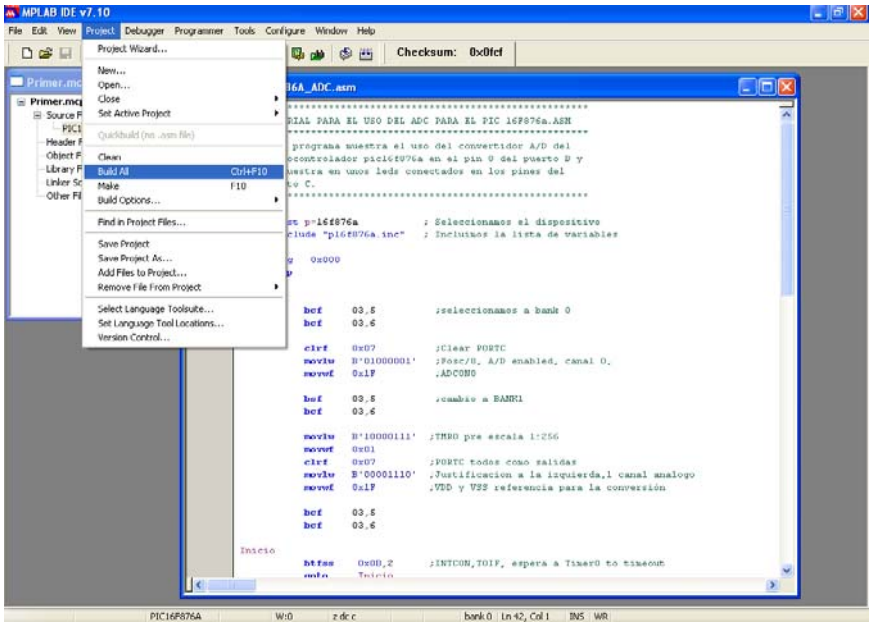


Fig. 6.14. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. Para compilar en el menú de Project se selecciona build all o F10 para compilar el programa previamente echo.

Sí no hay mensajes de error el programa fue exitoso.

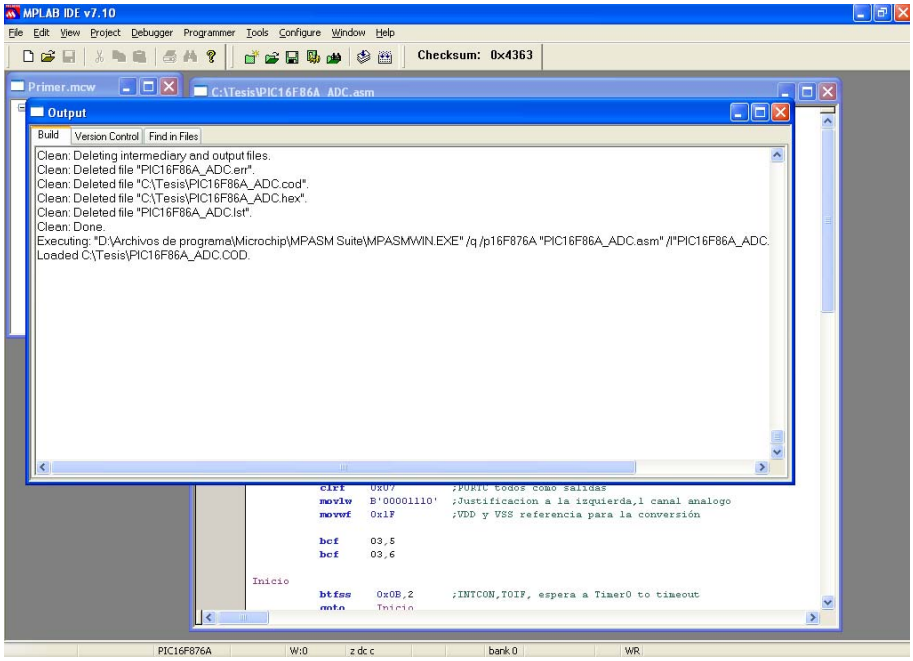


Fig. 6.15. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. En esta imagen se muestra que el programa fue ¡exitoso!

Conectamos el programador PIC Start plus

Fig. 6.16. Imagen donde se muestra el programador Pic Start Plus. Este programador solo funciona directo del programador MPLAB.

Activar el programador desde MPlab

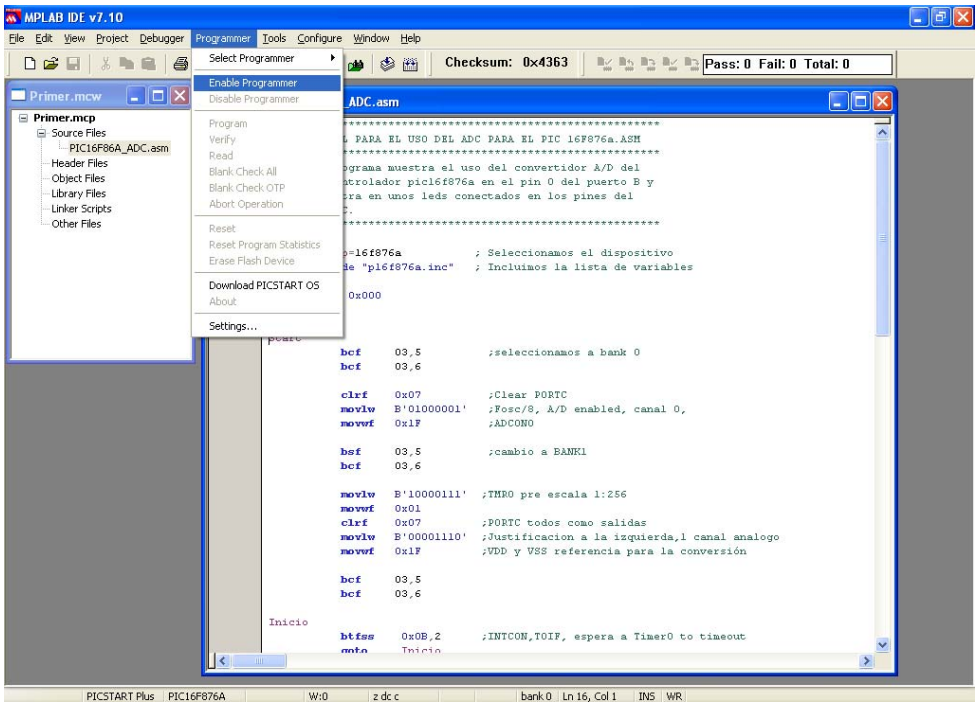


Fig. 6.17. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. En el menú de Enable Programer damos clic y a continuación se muestra un texto donde nos dice que el encendido del programador fue ¡exitoso!

Con el programador encendido podemos ya programar el Microcontrolador

Una vez que se selecciono en el menú Start program en el programador se activa el led rojo que nos indica que esta programando acabado de programarse el chip se apaga el led y nos dice en pantalla que la programación fue exitosa!

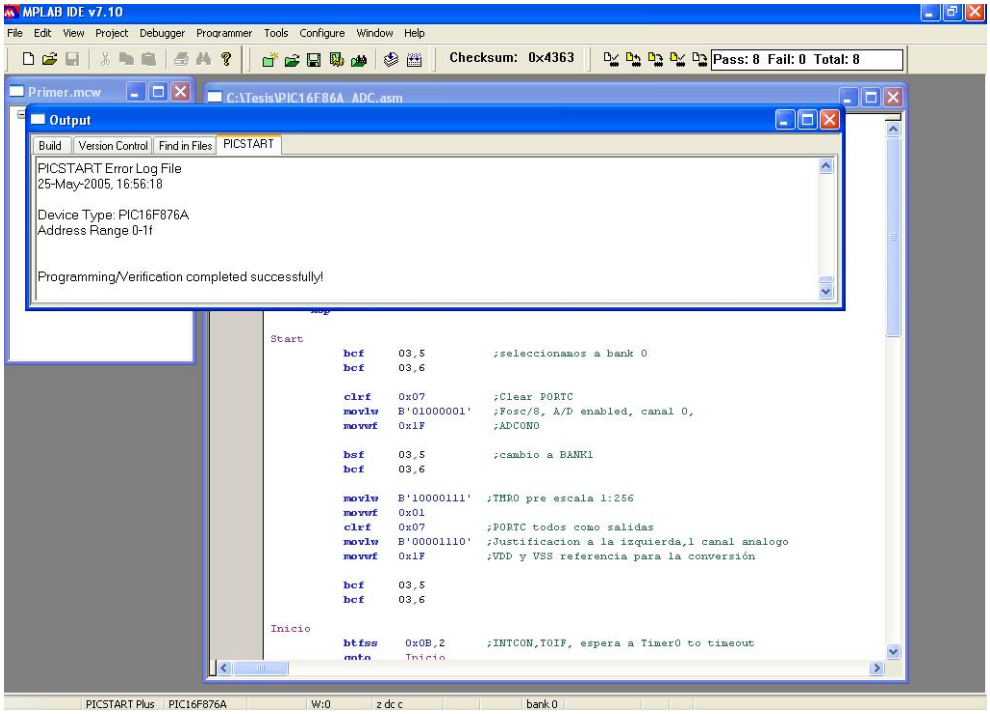


Fig. 6.18. Imagen capturada del programa MPLAB IDE V7.10 donde se muestra pasó a paso la programación de un microcontrolador. En esta imagen se muestra que la programación en el chip fue ¡exitosa!

UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO

Ejemplos didácticos con mini robots VII

Introducción

ROBÓTICA.

El término robótica procede de la palabra robot. La robótica es, por lo tanto, la ciencia o rama de la ciencia que se ocupa del estudio, desarrollo y aplicaciones de los robots.
Otra definición de robótica es el diseño, fabricación y utilización de máquinas automáticas programables con el fin de realizar tareas repetitivas como el ensamble de automóviles, aparatos, etc. y otras actividades. Básicamente, la robótica se ocupa de todo lo concerniente a los robots, lo cual incluye el control de motores, mecanismos automáticos neumáticos, censores, sistemas de cómputos, etc.

La robótica es una disciplina, con sus propios problemas, sus fundamentos y sus leyes. Tiene dos vertientes: teórica y práctica. En el aspecto teórico se aúnan las aportaciones de la automática, la informática y la inteligencia artificial. Por el lado práctico o tecnológico hay aspectos de construcción (mecánica, electrónica), y de gestión (control, programación). La robótica presenta por lo tanto un marcado carácter interdisciplinario.

En la robótica se aúnan para un mismo fin varias disciplinas afines, pero diferentes, como la Mecánica, la Electrónica, la Automática, la Informática, etc. El término robótica se le atribuye a Isaac Asimov. Los tres principios o leyes de la robótica según Asimov son:

Un robot no puede lastimar ni permitir que sea lastimado ningún ser humano.

El robot debe obedecer a todas las órdenes de los humanos, excepto las que contraigan la primera ley.

El robot debe autoprotegerse, salvo que para hacerlo entre en conflicto con la primera o segunda ley.

Primer Robot

Dos GAL 22v10
4 Temporizadores NE555
2 servomotores
Un Display grafico 16x2 con luz de fondo Figura. 7.1
Un puente en H LB293B
Un Inversor con Tigger Smitch 74LS14
Dos Infrarrojos Emisores de largo alcance 14°. Figura 7.4
Dos Receptor de Infrarrojos con filtro de luz de día. Figura. 7.4
2 servomotores. Figura 7.2
Tablas para soldar modelo 505 y 405
Un carrito lego

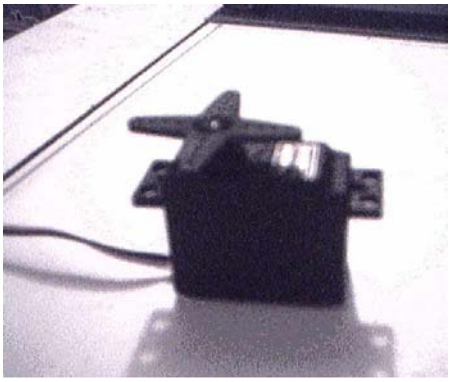
Fig. 7.1 Display grafico 16x2 con luz de fondo.

Fig. 7.2. Servomotor Futaba.

Fig. 7.3. Carrito lego.

Fig. 7.4. Emisor y receptor infrarrojo.

Ahora que ya sabemos como modificar un servomotor para que gire infinitamente, esta listo y ahora tomamos por un juguete lego para tomar de ahí las piezas que nos sirvan especialmente las llantas que sujetaremos a nuestro servo

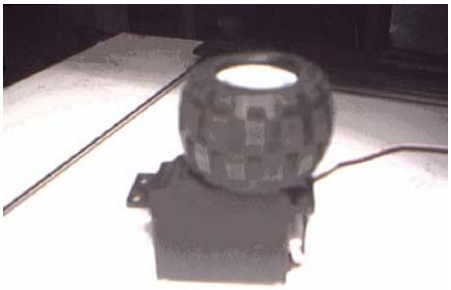


1. Tomamos el servomotor con la estrella colocada y recortamos las puntas hasta el 3 orificio para después atornillar la rudeda

Fig. 7.5. Servomotor con estrella colocada.

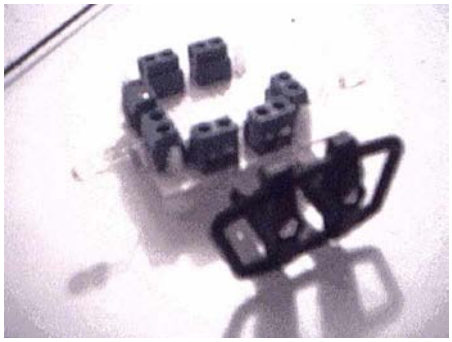
2. Nos fijamos que la rueda quede perfectamente alineada

Fig. 7.6. Servomotor con rin colocado.



3. Pasamos a colocar la goma en la llanta y verificar que la rueda gire perfectamente.

Fig. 7.7. Servomotor con rin colocado.



El tumba burros es ideal para insertarle dos LEDs trasparentes que prenden blanco. Esto lo montamos con terminales con tornillo en una placa base modelo 505 de esteren. Figura 7.8.

Fig. 7.8. Tumba Burros, leds y placa base instalados

Ahora colocamos los servos contra puestos para tener las llantas frontales pegadas con una cinta de doble adhesivo Fig 7.9.

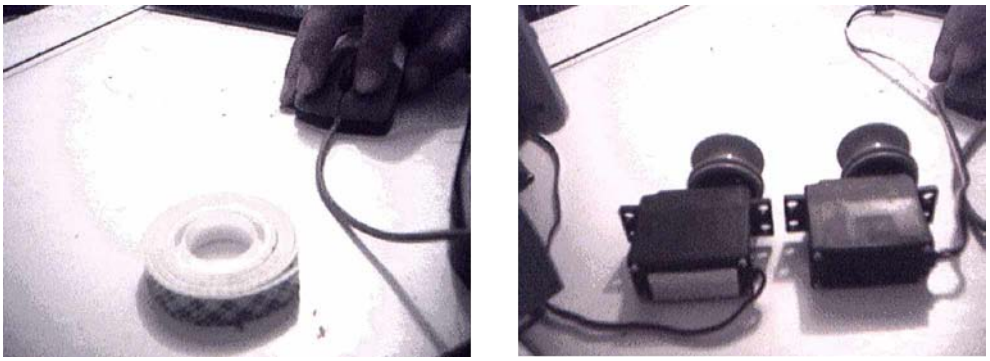
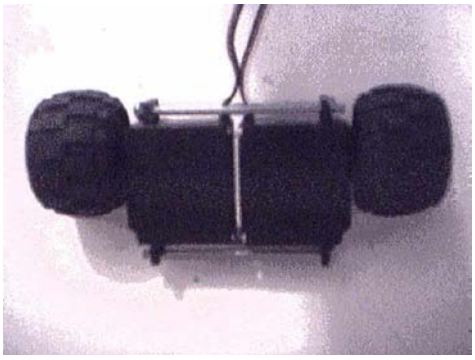
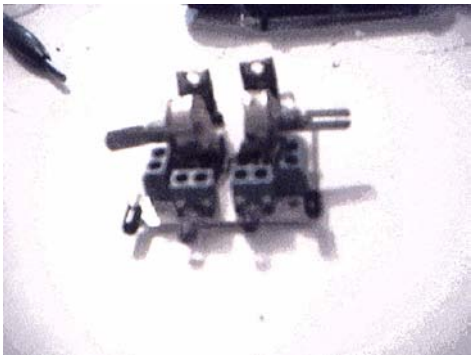


Fig. 7.9. Adhesivo de dos caras y servomotores.



Pasamos a unirlos con 4 tornillos de 1/8 de diámetro y colocar sus respectivas gomas. Fig 7.10.

Fig. 7.10. Servomotores encontrados opuestamente.



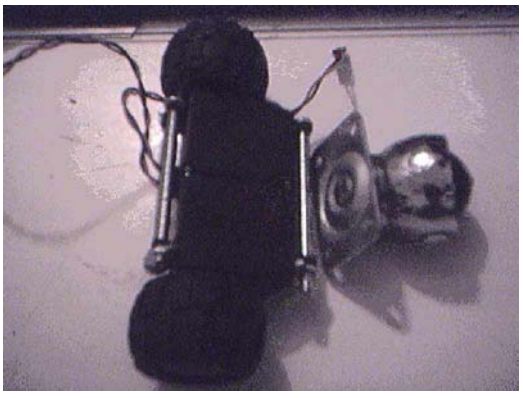
Ahora haremos el circuito infrarrojo que nos dirá si nuestro robot se ha salido de la línea blanca o negra. Figura 7.11.

Fig. 7.11. Censores infrarrojos.

Pasamos al programa en una Gal20v10

Como nuestros motores están contrapuestos para lo que a uno es adelante para el otro es atrás así que necesitamos un decodificador, un inversor y un multiplexor, por separado podremos hacerlo pero es mucho material y espacio y precisamente esa es la función de nuestra Gal22V10, para ahorrarnos todo ese enredo.

Estado infrarrojo	Entrada A	Entrada B	Entrada C	Salida 1 Motor izquierdo	Salida 2 Motor derecho	Focos frontales Altas	Focos frontales Bajas	Focos traseros noche	Reversa	Freno	Foco Lateral Izquierdo Frontal y tracero intermitente	Foco Lateral derecho Frontal y tracero intermitente
00	Pulso Bajo	Pulso Medio	Pulso Alto	A	C	1	1	1	0	0	0	0
01	Pulso Bajo	Pulso Medio	Pulso Alto	A	B	0	1	1	0	1	0	1
10	Pulso Bajo	Pulso Medio	Pulso Alto	B	C	0	1	1	0	1	1	0
	Pulso Bajo	Pulso Medio	Pulso Alto	C	A	0	1	1	1	0	0	0



Ahora pondremos nuestra colita que nos servirá de apoyo en la parte trasera. Fig 7.13.

Fig. 7.12. Servomotores encontrados opuestamente.

Alambramos correspondientemente las terminales y listo aquí tenemos nuestro robot seguidor de línea negra. Figuras

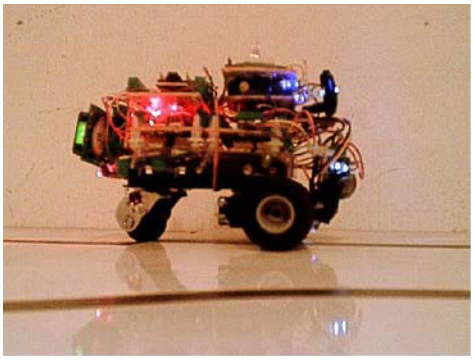


Fig. 7.13. Vista lateral izquierda del mini robot .

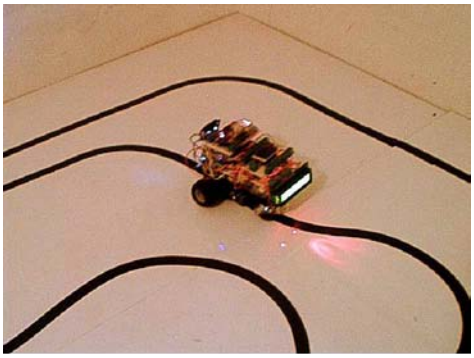


Fig. 7.14. Vista superior del mini robot .

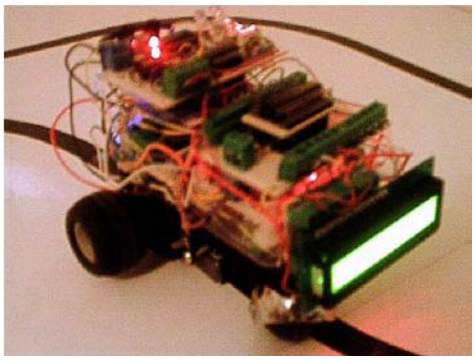


Fig. 7.15. Vista trasera lateral

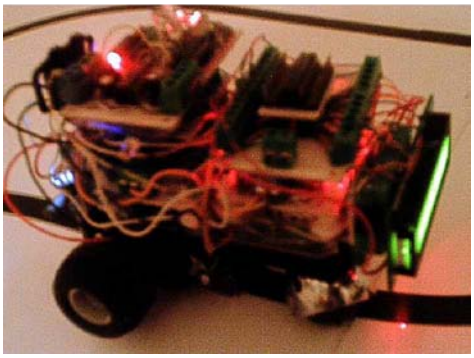
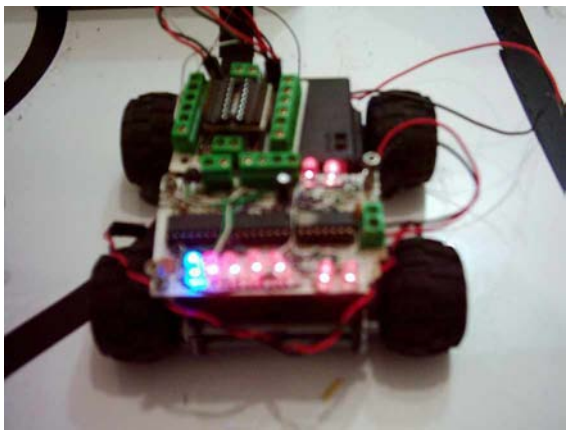


Fig. 7.16. Vista lateral derecha

Segundo mini robot

Controlado por infrarrojo PIC16F876A.



Este mini robot es controlado por infrarrojo como anteriormente hicimos el circuito ahora le damos un uso.

Fig. 7.17. Vista frontal del mini robot

Este mini robot es controlado por infrarrojos a distancia, con el circuito anterior, y un programa que explico continuación. El programa para este mini robot se muestra en la figura 7.18

File Edit Search Options View Help

Carrio_Infrarrojo.bas | Carrito_por_radio.bas

```
  DEVICE 16F877a                  ' Usamos el pic 16F876a

    CLS                           ' Limpiamos el LCD
    PORTA = 0                     ' PORTA en bajo para leer los botones
    TRISA = %00001111             ' Los pines del puerto A como entradas
    TRISD = %00000000             ' Los pines del puerto C como salidas


Inicio:
'           D               C               B               A           76543210              1234567890123456
 IF PORTA.0=1 Then IF PortA.1=1 Then IF PortA.2=1 Then IF PortA.3=1 then PortD=0000000: Print at 1,1, "Parado          "
 IF PORTA.0=1 Then IF PortA.1=1 Then IF PortA.2=1 Then IF PortA.3=0 then PortD=0000101: Print at 1,1, "Enfrente        "
 IF PORTA.0=1 Then IF PortA.1=1 Then IF PortA.2=0 Then IF PortA.3=1 then PortD=0000110: Print at 1,1, "Derecha         "
 IF PORTA.0=1 Then IF PortA.1=1 Then IF PortA.2=0 Then IF PortA.3=0 then PortD=0000100: Print at 1,1, "Avanza y Derecha"
 IF PORTA.0=1 Then IF PortA.1=0 Then IF PortA.2=1 Then IF PortA.3=1 then PortD=0001010: Print at 1,1, "Retrocede       "
 IF PORTA.0=1 Then IF PortA.1=0 Then IF PortA.2=1 Then IF PortA.3=0 then PortD=0000000: Print at 1,1, "Parado          "
 IF PORTA.0=1 Then IF PortA.1=0 Then IF PortA.2=0 Then IF PortA.3=1 then PortD=0000010: Print at 1,1, "Retro y derecha "
 IF PORTA.0=1 Then IF PortA.1=0 Then IF PortA.2=0 Then IF PortA.3=0 then PortD=0000000: Print at 1,1, "Parado          "
 IF PORTA.0=0 Then IF PortA.1=1 Then IF PortA.2=1 Then IF PortA.3=1 then PortD=0001001: Print at 1,1, "Izquierda       "
 IF PORTA.0=0 Then IF PortA.1=1 Then IF PortA.2=1 Then IF PortA.3=0 then PortD=0000001: Print at 1,1, "Avanza e izquier"
 IF PORTA.0=0 Then IF PortA.1=1 Then IF PortA.2=0 Then IF PortA.3=1 then PortD=0000000: Print at 1,1, "Parado          "
 IF PORTA.0=0 Then IF PortA.1=1 Then IF PortA.2=0 Then IF PortA.3=0 then PortD=0000000: Print at 1,1, "Parado          "
 IF PORTA.0=0 Then IF PortA.1=0 Then IF PortA.2=1 Then IF PortA.3=1 then PortD=0001000: Print at 1,1, "Retro e izquierd"
 IF PORTA.0=0 Then IF PortA.1=0 Then IF PortA.2=1 Then IF PortA.3=0 then PortD=0000000: Print at 1,1, "Parado          "
 IF PORTA.0=0 Then IF PortA.1=0 Then IF PortA.2=0 Then IF PortA.3=1 then PortD=0000000: Print at 1,1, "Parado          "
 IF PORTA.0=0 Then IF PortA.1=0 Then IF PortA.2=0 Then IF PortA.3=0 then PortD=0000000: Print at 1,1, "Parado          "

GOTO Inicio
END
```

Fig. 7.18. Programa para el segundo mini robot

Para cada valor en la entrada del PIC y que es dado por el receptor de infrarrojos tenemos una salida que va a la entrada del puente en H del chip L293D, cuando en el emisor pulsamos una entrada en alto, en la salida correspondiente del receptor se va a nivel bajo. Por eso colocamos en el programa un orden al revés de cómo normalmente estaría si se activaran de bajo a alto.

Tercer mini robot

Controlado por PIC 18F452

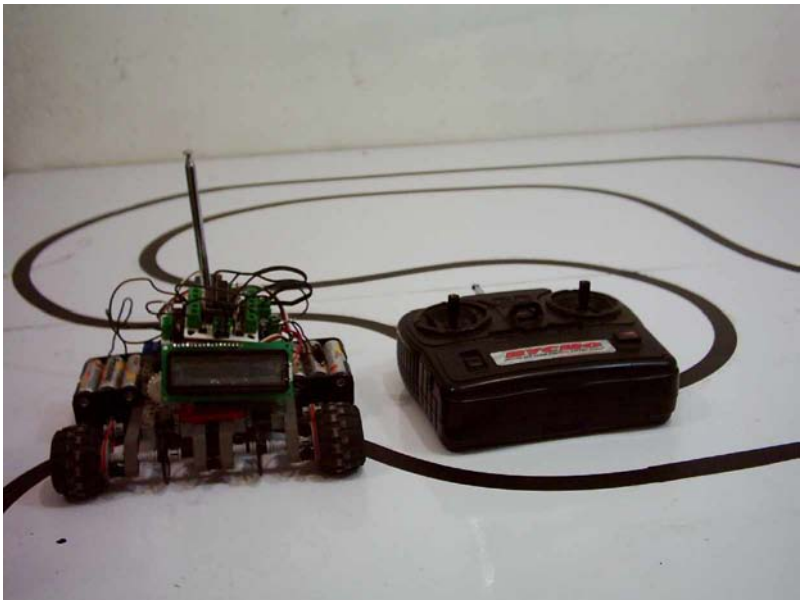


Fig. 7.21. Vista frontal del mini robot

Este mini robot es controlado por un equipo estándar de radio control de dos canales Figura 7.21.

El primer canal del radio control que nos dice que el mini robot avanza o retrocede realmente esta conectado a un servomotor que gira una resistencia variable en los carros de radio control eléctricos para determinar si avanza o retrocede con cierta velocidad, en los autos de radio control por motor a gasolina, el servomotor tira de un chicote para hacer acelerar el motor. Pero con un microcontrolador se puede controlar mejor esta variable. Con una característica llamada modulación de ancho de pulsos (PWM).

El PWM como su nombre lo indica modula el ancho de pulso de una frecuencia X. esta característica podemos usarla para hacer variar la duración del estado alto y así controlar en valor eficaz de una frecuencia X.

PWM y HPWM

En un microcontrolador podemos hacer un PWM con simplemente hacer una comparación. Si el número en binario en la entrada de un puerto del microcontrolador es mayor a la que esta en un registro que aumentamos en 1 en cada ciclo del programa coloca en 0 la salida que en el inicio del programa estaba en 1, pero este programa estaría siempre encargado controlar el PWM, pero algunos microcontroladores tienen el PWM como interrupción, esto nos deja que el microcontrolador haga otras funciones mientras hace a su vez un PWM. El PIC 18F452 tiene dos interrupciones para un HPWM.

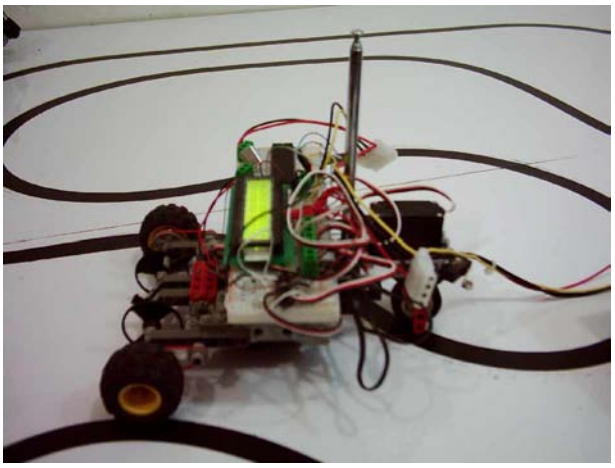


Fig. 7.22. Vista lateral de tercer mini robot

Descripción del programa para este mini robot

Escogemos el PIC 18F452
Declaramos CCP1_PIN en los pines del puerto C1 y CCP2_PIN C2
Declaramos 3 variable del tipo Word una para almacenar los datos de la lectura del receptor de radio control y otras dos para el valor del Hardware PWM.
Todas las salidas y entradas como digitales
Limpiamos el LCD
Las variables en 0
Comienza el Loop
Las variables en 0
Hacemos la lectura del receptor del radio control en el pin 0 del puerto C.
Para cada valor que recibimos del receptor tenemos una tabla equivalente a HPWM
Sí esta en X valor manda X valor en HPWM a la salida al pin C1 y C2
Y en el display muestra el valor leído y valor de salida.
Fin del programa
Y así hacemos que retroceda o avance el mini robot de radio control.
El otro canal va directo al servomotor.

Programa.

C:\Carrito_por_radio.bas

File Edit Search Options View Help

Carrito_por_radio.bas

```
Device 18F452
Xtal 4

DECLARE CCP1_PIN PORTC.1
DECLARE CCP2_PIN PORTC.2

DIM VAR1 as WORD

DIM POS1 as WORD
DIM POS2 as WORD

ALL_DIGITAL = TRUE

        POS1=0
        POS2=0
        VAR1=0


Cls
Loop:
       POS1=0
       POS2=0

       VAR1 = PULSIN PORTC.0 , 1     ' Measure a pulse on pin 0 of PORTB.


       If VAR1  = 148 then  POS1=8
       If VAR1  = 147 then  POS1=16
       If VAR1  = 146 then  POS1=24
       If VAR1  = 145 then  POS1=32
       If VAR1  = 144 then  POS1=40
       If VAR1  = 143 then  POS1=48
       If VAR1  = 142 then  POS1=56
       If VAR1  = 141 then  POS1=64
       If VAR1  = 140 then  POS1=72
       If VAR1  = 139 then  POS1=80
       If VAR1  = 138 then  POS1=88
       If VAR1  = 137 then  POS1=96
       If VAR1  = 136 then  POS1=104
       If VAR1  = 135 then  POS1=112
       If VAR1  = 134 then  POS1=120
       If VAR1  = 133 then  POS1=128
       If VAR1  = 132 then  POS1=136
       If VAR1  = 131 then  POS1=144
       If VAR1  = 130 then  POS1=152
       If VAR1  = 129 then  POS1=160
       If VAR1  = 128 then  POS1=168
       If VAR1  = 127 then  POS1=176
       If VAR1  = 126 then  POS1=184
       If VAR1  = 125 then  POS1=192
       If VAR1  = 124 then  POS1=200
       If VAR1  = 123 then  POS1=208
       If VAR1  = 122 then  POS1=216
       If VAR1  = 121 then  POS1=224
       If VAR1  = 120 then  POS1=232
       If VAR1  = 119 then  POS1=240
       If VAR1  = 118 then  POS1=248
       If VAR1 <= 117 then  POS1=255
```

```
if VAR1  = 153 then  POS2=8
if VAR1  = 154 then  POS2=16
if VAR1  = 155 then  POS2=24
if VAR1  = 156 then  POS2=32
if VAR1  = 157 then  POS2=40
if VAR1  = 158 then  POS2=48
if VAR1  = 159 then  POS2=56
if VAR1  = 160 then  POS2=64
if VAR1  = 161 then  POS2=72
if VAR1  = 162 then  POS2=80
if VAR1  = 163 then  POS2=88
if VAR1  = 164 then  POS2=96
if VAR1  = 165 then  POS2=104
if VAR1  = 166 then  POS2=112
if VAR1  = 167 then  POS2=120
if VAR1  = 168 then  POS2=128
if VAR1  = 169 then  POS2=132
if VAR1  = 170 then  POS2=140
if VAR1  = 171 then  POS2=148
if VAR1  = 172 then  POS2=156
if VAR1  = 173 then  POS2=164
if VAR1  = 174 then  POS2=172
if VAR1  = 175 then  POS2=180
if VAR1  = 176 then  POS2=188
if VAR1  = 177 then  POS2=196
if VAR1  = 178 then  POS2=204
if VAR1  = 179 then  POS2=216
if VAR1  = 180 then  POS2=224
if VAR1  = 181 then  POS2=232
if VAR1  = 182 then  POS2=240
if VAR1  = 183 then  POS2=248
if VAR1 >= 184 then  POS2=255



Print AT 1,1, "N1 ",@VAR1, " S1 ",@POS1," "
Print AT 2,1, "N2 ",@VAR1, " S2 ",@POS2," "

HPWM 1,POS1,32767
HPWM 2,POS2,32767

GOTO Loop                ' Repeat the process.
end
```

Conclusiones.

Hacer un temporizador con un 555 para cada servomotor es muy tedioso pero si se conecta directo a un radio control esto lo hace girar infinitamente. En el siguiente mini robot se conecta directamente un voltaje a los motores.

Con el programa y el circuito es fácil repetirlo para cualquier interesado en este experimento, a lo que quiero llegar, es que lo enseñado en la escuela puede ser aplicado y que se vea en un ejemplo real. Este control es bueno solamente cuando se tiene el objeto a controlar a la vista pues no atraviesa paredes para esto en el siguiente mini robot es controlado por radio y montada una mini cámara inalámbrica para saber donde esta el objeto que no vemos pero si sabemos lo que se ve a través de el.

Fig. 8.1. Vista frontal del mini robot

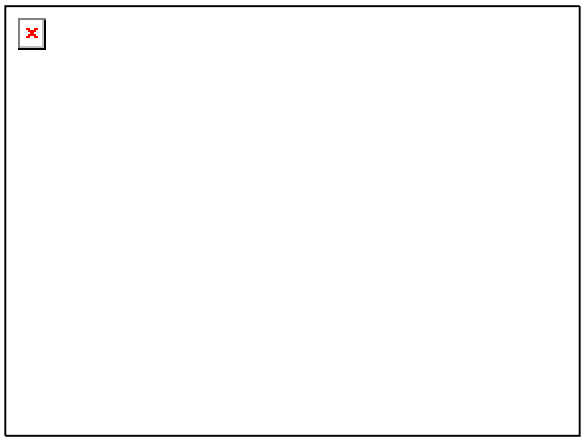
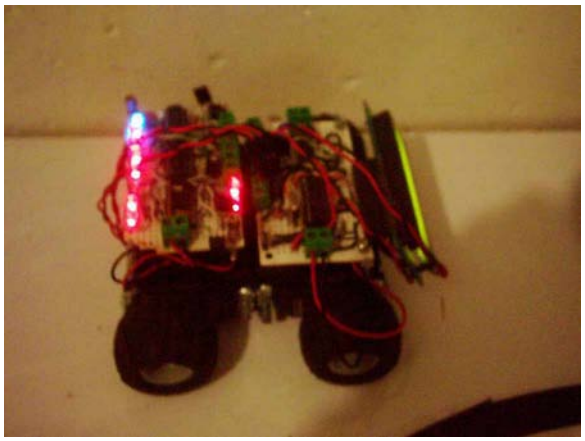


Fig. 7.20. Vista lateral del segundo mini robot



Teniendo el programa se puede usar para múltiples usos y eliminamos las partes móviles en los sistemas donde un servomotor tenía que girar una resistencia variable para determinar si avanzaba o retrocedía con cierta potencia y eliminamos las perdidas disipadas en la resistencia.

Bibliografías

1. - Pathfinder, El robot, Editorial: F&G editores 2002 S.A. España.
2. - www.ag-electronica.com Manuales hojas de datos PDF
3. - www.microchip.com Mid Range MCU Family Reference Manual
4.-http://www.geocities.com/SiliconValley/Circuit/8882/djlcdsim/djlcdsim.html simulador.
5.- www.cypress.com. Manuales y hojas de datos.
6.- www.tanya.com. Manuales y hojas de datos para modulo LCD 16x2 y LCD 128x64
7.- www.futaba.com. Manuales y hojas de datos para servomotor modelo 3004
8.- www.letbasic.com Manual de referencia
9.- www.parallax.com Manual de referencia.
10.- www.pic-basic.de Manual de referencia.
11.- www.picbasic.org Manual de referencia.
12.- www.crownhill.com Manual de referencia.

Anexos

Apéndice

Circuitos lógicos

Un circuito lógico es aquel que maneja la información en forma de "1" y "0", dos niveles de voltaje fijos. "1" nivel alto y "0" nivel bajo.

Estos circuitos están compuestos por elementos digitales como las compuertas: AND (Y), OR (O), NOT (NO).

y combinaciones poco o muy complejas de estos. Estas combinaciones dan lugar a otros tipos de elementos digitales como los compuertas, entre otros.

- nand (No Y)
- nor (No O)
- or exclusiva (O exclusiva)
- mutiplexores o multiplexadores
- demultiplexores o demultiplexadores
- decodificadores
- codificadores
- memorias
- flip-flops
- microprocesadores
- microcontroladores
- etc.

La electrónica moderna usa electrónica digital para realizar muchas funciones. Aunque los circuitos electrónicos pueden resultar muy complejos, en realidad se construyen de un número muy grande de circuitos muy simples.

En un circuito digital se transmite información binaria (ceros y unos) entre estos circuitos y se consigue un circuito complejo con la combinación de bloques de circuitos simples.

La información binaria se representa en la forma de "0" y "1", un interruptor "abierto" o "cerrado", "On" y "Off", "falso" o "verdadero", en donde "0" representa falso y "1" verdadero.

Los circuitos lógicos se pueden representar de muchas maneras. En los circuitos siguientes la lámpara puede estar encendida o apagada ("on" o "off"), dependiendo de la posición del interruptor. (apagado o encendido).

Los posibles estados del interruptor o interruptores que afectan un circuito se pueden representar en una tabla de verdad. Las tablas de verdad pueden tener muchas columnas, pero todas las tablas funcionan de igual forma. Hay siempre una columna de salida que representa el resultado de todas las posibles combinaciones de las entradas.

Tabla de verdad	
Columna(s) de entrada	**Columna de salida**
Entrada (interruptor)	Salida (lámpara)
Abierto	Apagado
Cerrado	Encendido

El Número de columnas en una tabla de verdad depende de cuantas entradas hay + 1 (la columna de la salida), el número de filas representa la cantidad de combinaciones en las entradas.

Número de combinaciones = 2^n, donde n es el número de columnas de la tabla de verdad (menos la columna de salida).

Ejemplo: en la siguiente tabla hay 3 columnas de entrada, entonces habrán: $2^3 = 8$ combinaciones (8 filas).

Un circuito con 3 interruptores de entrada (con estados binarios "0" o "1"), tendrá 8 posibles combinaciones. Siendo el resultado (la columna salida) determinado por el estado de los interruptores de entrada.

Tabla de verdad			
Switch 1	**Switch 2**	**Switch 3**	**Salida**
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

Los circuitos lógicos son básicamente un arreglo de interruptores, conocidos como "compuertas lógicas" (compuertas AND, NAND, OR, NOR, NOT, etc.) Cada compuerta lógica tiene su tabla de verdad. Y, si pudiéramos ver en mas detalle la construcción de éstas, veríamos que es un circuito comprendido por transistores, resistencias, diodos, etc. conectados de manera que se obtienen salidas específicas para entradas específicas.

La utilización extendida de las compuertas lógicas, simplifica el diseño y análisis de circuitos complejos. La tecnología moderna actual permite la construcción de circuitos integrados (IC´s) que se componen de miles (o millones) de compuertas lógicas.

Complemento

Compuertas lógicas

Compuerta No

Dentro de la electrónica digital, no se podrían lograr muchas cosas si no existiera la compuerta NOT (compuerta NO), también llamada compuerta inversora, que al igual que las compuertas AND y OR tiene una importancia fundamental.

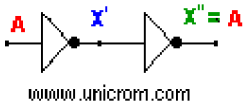
La compuerta NOT se representa con el siguiente símbolo, y su tabla de verdad es:

A	X
0	1
1	0

La salida de una compuerta "NOT" tiene el valor inverso al de su entrada. En el caso del gráfico anterior la salida X = A. Esto significa que si a la entrada tenemos un "1" lógico, a la salida hará un "0" lógico y si a la entrada tenemos un "0" a la salida habrá un "1".

Nota: El apóstrofe en la siguiente expresión significa "negado": X = A' y es igual a X = A.

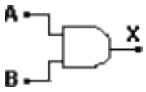
La compuertas NOT se pueden conectar en cascada, logrando después de dos compuertas, la entrada original. Ver la siguiente Fig.



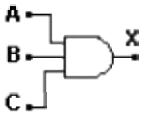
A	X'	X''
0	1	0
1	0	1

Compuerta Y

Es una de las compuertas mas simples dentro de la Electrónica Digital. Su representación es la que se muestra en las figuras. Como se puede ver tiene dos entradas A y B, aunque puede tener muchas más (A,B,C, etc.) y sólo tiene una salida X.



compuerta AND de 2 entradas



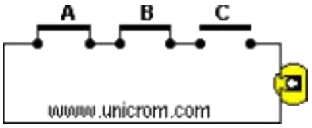
compuerta AND de 3 entradas

La compuerta AND de 2 entradas tiene la siguiente tabla de verdad

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Se puede ver claramente que la salida X solamente es "1" (1 lógico, nivel alto) cuando tanto la entrada A como la entrada B están en "1".En otras palabras "La salida X es igual a 1 cuando la entrada A y la entrada B son 1

Esta situación se representa en el álgebra booleana como: X = A * B o X = AB. Una compuerta AND de 3 entradas se puede implementar con interruptores de la siguiente manera:



A	B	C	Lámpara
A	A	A	Apagada
A	A	C	Apagada
A	C	A	Apagada
A	C	C	Apagada
C	A	A	Apagada
C	A	C	Apagada
C	C	A	Apagada
C	C	C	Encendida

A = Abierto C = Cerrado

Una compuerta AND puede tener muchas entradas. Una AND de múltiples entradas puede ser creada conectando compuertas simples en serie. Si si se necesita una AND de 3 entradas y no hay disponible, es fácil crearla con dos compuertas AND en serie o cascada como se muestra en la siguiente figura:

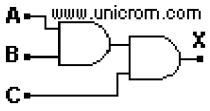


Tabla de verdad			
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

De igual manera, se puede implementar circuitos AND de 4 o más entradas.

Compuerta O

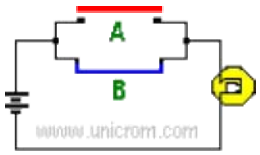
Es una de las compuertas mas simples dentro de la Electrónica Digital.. Su representación y tabla de verdad se muestran a continuación:

Tabla de verdad

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Y se representa con la siguiente función booleana: X = A + B o X = B + A

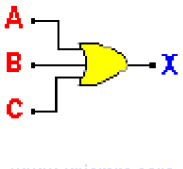
Esta misma compuerta se puede implementar con interruptores como se muestra en la siguiente figura, en donde se puede ver que: cerrando el interruptor A "O" el interruptor B se encenderá la luz.



"1" = cerrado , "0" = abierto, "1" = luz encendida

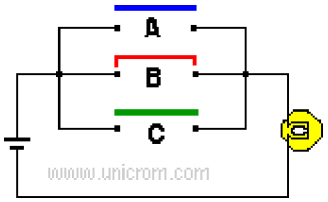
En las siguientes figuras se muestran la representación de la compuerta "OR" de tres entradas con su tabla de verdad y la implementación con interruptores

Representación de una compuerta OR de 3 entradas con su tabla de verdad



A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Compuerta "OR" de 3 entradas implementada con interruptores



Se puede ver claramente que la luz se encenderá cuando cualquiera: A o B o C este cerrada

Glosario

Servomotor - Un servomotor es un dispositivo electromecánico en el cual una retroalimentación determina la posición del rotor.

PWM – Pulse Wave Modulation, modulación de la onda pulso, modulación de ancho de pulso.

Dip – Tipo de encapsulado de los circuitos electrónicos.

Dual in line package – Tipo de encapsulado de los circuitos doble tamaño de ancho comparado con dip.

PLCC – Tipo de encapsulado de los circuitos electrónicos cuadrado de 44 pines.

Display – Pantalla echa con cristal líquido.

DB25 – Tipo de terminal para conexiones a 25 líneas a una computadora.

Carácter – letra que es conformada por un conjunto de 8 bits ó un byte.