UACM

Universidad Autónoma de la Ciudad de México

Nada humano me es ajeno

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

Diseño, implementación y aplicaciones de un bootloader para microcontroladores

TRABAJO RECEPCIONAL

QUE PARA OBTENER EL TÍTULO DE LICENCIADO EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

PRESENTA

ERNESTO DAVID ROJAS SERRANO

DIRECTOR DEL TRABAJO

Dr. Daniel Noriega Pineda

Ciudad de México, agosto de 2017.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

Diseño, implementación y aplicaciones de un bootloader para microcontroladores.

ERNESTO DAVID ROJAS SERRANO

Agosto 2017

Agradecimientos

A mis padres, familiares y amigos, en especial a mi hermana que ha sido mi inspiración.

Índice general

Índice de figuras

Capítulo 1

Introducción

En la actualidad lo más probable es que encontremos un procesador/microcontrolador en la mayoría de los dispositivos que utilizamos a diario, por ejemplo teléfonos inteligentes, microondas, pantallas de televisión, computadoras, refrigeradores, etc.

Todos estos dispositivos incluyen características iniciales, para el caso de un teléfono inteligente podría ser un sistema operativo con versiones actualizables, para el caso de una pantalla de televisión son las aplicaciones que permiten acceder a diversos servicios de streaming, y para el caso de una computadora sería el firmware del bios.

Aquí es importante hacer notar que el mercado se rige generalmente por dos factores, por un lado se encuentra al factor económico y por otro lado el factor de la innovación, estos factores influyen en el producto final que es vendido al consumidor, por parte del factor económico este está sujeto a un presupuesto de desarrollo, este presupuesto muchas veces es muy ajustado, por lo tanto no se puede invertir más allá de lo que se tiene calculado, aun cuando el margen de ganancia es amplio, como consecuencia se lanza al mercado dispositivos que no se encuentran debidamente optimizados, ahora por parte del factor innovación este se debe a la rapidez en la cual el dispositivo debe ser lanzado al mercado, dando nuevamente como consecuencia un dispositivo que no ha sido optimizado.

Dicho de otra forma, estos dispositivos vienen con versiones de firmware poco optimizadas o con errores de programación, y debido a que a diario se diseñan y fabrican nuevos dispositivos los fabricantes necesitan actualizar sus dispositivos, es aquí donde los bootloaders son utilizados para solventar esos problemas:

La ventaja de un bootloader se puede resumir en dos partes:

- Permite corregir errores.
- Permite añadir nuevas características.

Corregir errores

Un bootloader se encarga de re-programar el firmware del dispositivo, esto con el fin de corregir los errores de programación/optimización que pueda contener.

Un ejemplo podría ser mejorar la respuesta de la pantalla táctil de un teléfono inteligente, o bien mejorar el consumo energético del dispositivo, esto por lo general se debe a la baja optimización del código que controla el dispositivo.

Añadir nuevas características

Un bootloader es capaz de actualizar el firmware con el objetivo de soportar nuevos dispositivos, o bien puede ser utilizado para añadir nuevas características de hardware que no estaban implementadas en un inicio.

Por lo general, los dispositivos están diseñados para poder soportar futuras tecnologías, un ejemplo sería soportar el formato 4k en la cámara del teléfono inteligente o bien poder desbloquear el teléfono utilizando la cámara frontal.

1.1. Objetivo general

Diseñar un bootloader para microcontroladores y proponer diversos escenarios donde puede ser utilizado.

1.2. Objetivos específicos

- Crear una plantilla genérica para el diseño de un bootloader.

- Demostrar las ventajas que presenta un dispositivo con un bootloader frente a dispositivos con programación permanente.

Capítulo 2

Antecedentes

En el siguiente capítulo se explicara de manera general que es un bootloader, de que consta, algunas aplicaciones y precauciones a tomar en cuenta.

2.1. ¿Qué es un microcontrolador?

Un microcontrolador es un circuito integrado programable, donde en su interior posee los mismos elementos que una computadora digital (unidad central de procesamiento, memoria y dispositivos de entrada-salida) y opcionalmente algunos dispositivos periféricos como convertidores analógicos-digitales, digitales-analógicos, circuitos generadores de modulación por ancho de pulso, puertos de comunicación serie, comparadores analógicos, entre otros, que auxilian al microcontrolador a cumplir la tarea para la que ha sido programado.

Un microcontrolador es un sistema cerrado porque todos los elementos que necesita para funcionar están dentro del mismo chip y no es posible modificar las características técnicas del dispositivo, a comparación de un sistema abierto al cual se le puede añadir nuevas características, un ejemplo de un sistema abierto sería una computadora, a este tipo de sistemas se le puede añadir más memoria RAM, más espacio en disco duro, tarjetas de video dedicadas, etc.

2.2. ¿Qué es un bootloader?

Un Bootloader es un segmento de código pre-programado que reside en la memoria programa de un microcontrolador, este código se ejecuta cada vez que el microcontrolador es energizado o cada vez que el microcontrolador se reinicia.

La función principal de un bootloader es re-programar la memoria de programa del microcontrolador, dicho de otra forma se encarga de modificar la programación principal del microcontrolador, por ejemplo supongamos que se ha programado un microcontrolador

para que prenda y apague un foco cada día a la misma hora, el programa principal se va a encargar de obedecer estas instrucciones, por lo tanto va a encargarse de encender y apagar el foco a la hora determinada, ahora bien, si el programador no tomó en cuenta que existe el horario de verano, el foco se va a encender una hora antes o una hora después de la hora que se desea, para solucionar este problema se puede utilizar el bootloader del microcontrolador para corregir la programación inicial y así indicarle que existe un cambio en la hora de prendido y apagado del foco, sin la necesidad de hardware especial o modificaciones físicas, el nuevo firmware puede ser transmitido desde internet o bien el usuario puede hacer esta actualización desde su computadora.

2.3. Aplicaciones de un bootloader

- Automatización

Sin importar cuál sea el proceso a controlar, el sistema de control debe ser flexible, tanto para añadir nuevas tareas, como para actualizar los parámetros de control que pudiera tener.

En sistemas basados en algoritmos de control convencionales se pueden actualizar los parámetros en línea, de manera que solo se actualizan las variables guardadas dentro de la memoria programa, la desventaja que esto presenta respecto a una re-programación total de la memoria se encuentra en la etapa de diseño, mientras que para el sistema que actualiza las variables internas se necesita un código adicional para manejar la información, el bootloader puede re-programar de manera rápida y eficiente el firmware lo cual se traduce en tiempos de diseño menores.

En sistemas de control flexibles, a los cuales se les puede añadir nueva tareas, no basta con poder modificar las variables internas ya que puede darse el caso que sea necesario añadir sensores adicionales, lo cual implica modificar el perfil de hardware del microcontrolador, por lo tanto es indispensable contar con un bootloader.

- Robótica

Este podría ser probablemente el campo donde la utilización de un bootloader es crítico, esto se debe al hecho de que el ajuste de los sensores y la programación de los algoritmos

se deben evaluar una vez que los módulos estén interconectados y situados en la estructura final del robot, puesto que se van a requerir ajustes posteriores la actualización del firmware va a ser continuo, por lo tanto se debe contar con un medio que sea capaz de re-programar la memoria programa de manera rápida, eficiente y sobre todo confiable, se podría utilizar ICSP/ISP (*Incircui Serial Programming/In-System Programming))* pero eso supone contar con hardware especializado para programar el microcontrolador, además de tiempo de diseño y costos adicionales, mientras que con la utilización de un bootloader se evitan estos inconvenientes.

- Internet de las cosas

En la actualidad los dispositivos tienden a ser dispositivos *inteligentes*, ya no es suficiente que un reloj de simplemente la hora, ahora es indispensable que pueda comunicarse con los demás dispositivos, aun cuando se traten de cosas opuestas, por ejemplo un teléfono celular o un refrigerador, esta es la propuesta del internet de las cosas o por sus siglas en inglés (IoT), básicamente se trata de una interconexión masiva de todos los dispositivos independientemente del tipo de dispositivo.

Aquí es donde es vital contar con un medio que sea capaz de añadir nuevas funciones a los dispositivos, de lo contrario su utilidad y tiempo de vida se ven comprometido, dicho de otra forma se vuelve obsoleto en muy poco tiempo.

2.4. Definición formal de un bootloader

En las primeras etapas del desarrollo de microcontroladores programar la memoria programa requería laboratorios y maquinas especializadas, con forme fue avanzando la tecnología se podía reprogramar estos chips mediante rayos ultra violetas, luego se dio un salto a memorias eléctricamente re-programables los cuales requerían de voltajes y secuencias muy estrictas para lograr re-programar la memoria, actualmente los microcontroladores cuentan con registros e instrucciones diseñadas para tal fin, esto significa que dentro del mismo encapsulado se cuenta con todo lo necesario para re-programar la memoria.

Un bootloader es un segmento de códigos que se encuentra alojado, ya sea en la parte baja o alta de la memoria de programa del microcontrolador, este segmento de código se encarga de atender una solicitud para re-programar la memoria de programa del microcontrolador, en caso contrario, si no se le hace dicha petición, el código debe ser capaz de apuntar hacia la posición de memoria donde reside el programa principal, este segmento de código se ejecuta cada vez que el dispositivo es energizado, o bien cada vez que se reinicia el microcontrolador.

Un programa (generalmente de computadora) se encarga de decodificar los datos del nuevo programa para posteriormente enviarlos al bootloader, dependiendo del bootloader, la forma en que éste recibe los datos puede ser mediante diferentes módulos de comunicación, los más comunes son UART, I2C, SPI y USB.

Por lo tanto un bootloader se conforma de dos partes:

- Programa servidor
- Programa cliente

Programa servidor.- Es el encargado de decodificar y enviar la cadena de valores que contiene las nuevas instrucciones a ser programadas dentro del programa principal.

Programa cliente.- Se encarga de recibir y programar las cadenas de valores dentro de la memoria programa del microcontrolador.

Para el caso de esta tesis el programa servidor es un programa que funciona bajo el sistema operativo windows, desarrollado en visual net, denominado JLoader, y un programa cliente desarrollado en lenguaje de bajo nivel identificado como JBoot.

2.5. Precauciones a tomar en cuenta

Como se mencionó anteriormente, el bootloader se va a ejecutar cuando se energiza o se reinicia el microcontrolador, el diseño del bootloader debe tomar en cuenta los factores que puedan reiniciar el microcontrolador, especialmente en aquellos casos donde se está controlando sistemas en tiempo real o en aquellos procesos donde el tiempo no es crítico.

Tabla de información sobre los diferentes registros que afectan el funcionamiento de un bootloader	
Registro	**Descripción**
POR	*Power-on reset* por sus siglas en ingles POR, mantiene el microcontrolador en estado de reset hasta que el voltaje de operación alcanza un nivel adecuado para la operación del circuito. Registro de activación: Palabra de configuración. Registro de configuración: No accesible a través de SFR.
BOR	*Brown-out reset* por sus siglas en ingles BOR, mantiene el microcontrolador en estado de reset si el voltaje de operación se encuentra por debajo del threshold establecido. Registro de activación: Palabra de configuración. Registro de configuración: No accesible a través de SFR.
WDT	*Watchdog timer* por sus siglas en ingles WDT, reinicia el microcontrolador cuando se alcanza el valor nominal del tiempo establecido. Registro de activación: Palabra de configuración. Registro de configuración: Accesible a través de SFR.
MCLR	*Master clear reset* por sus siglas en ingles MCLR, restablece los valores por defecto de todos los registros del microcontrolador, y apunta el contador de programa hacia el vector 0x00. Registro de activación: Palabra de configuración. Registro de configuración: No accesible a través de SFR.
Stack overflow	Causara un reset si ocurre un desbordamiento superior en la pila. Registro de activación: Palabra de configuración. Registro de configuración: Accesible a través de SFR.
Stack underflow	Causara un reset si ocurre un desbordamiento inferior en la pila. Registro de activación: Palabra de configuración. Registro de configuración: Accesible a través de SFR.

Cuadro 2.5. : *Registros importantes.*

El Cuadro 2.5.1 describe los factores que pueden reiniciar el microcontrolador.

Dependiendo del microcontrolador puede o no tener las características mencionadas; Por lo general en cualquier gama antigua (*legacy*) se puede encontrar las opciones MCLR, POR, BOR y WDR, mientras que en gamas de nueva generación para familias medias y altas se encuentran las características anteriores aparte de Stack overflow y Stack underflow.

Capítulo 3

Desarrollo JBoot:

El primer paso para el desarrollo de un bootloader es entender la arquitectura del microcontrolador, para el caso de los microcontroladores de las familias PIC12FXXXX, PIC16FXXXX y PIC18FXXXX de microchip nos encontramos con una arquitectura Hardvard, con registros de longitud de 8 bits y un set de instrucciones RISC.

- Arquitectura Hardvard

La arquitectura Hardvard tiene una memoria programa y una memoria dato separadas, a diferencia de la arquitectura Von Neumann (la cual junta la memoria programa y dato), por lo tanto la arquitectura Hardvard es más eficiente para el manejo de datos.

- RISC

Proviene de las siglas en inglés *reduced instruction set computer* (computadora con set de instrucciones reducido), esto significa que solo se requiere aprender un set de instrucciones no mayor de 30 para poder programarlo, a diferencia de un sistema CISC *complex instruction set computer* (computadora con set de instrucciones complejas) el cual puede llegar a tener decenas de instrucciones para una misma acción.

- Memoria de programa

En la Figura 3.1 se muestra la organización de la memoria programa, ésta organización no es exclusiva de un modelo de microcontrolador en específico, por lo tanto se puede utilizar como plantilla para el diseño de un bootloader para la familia 12F y 16F.

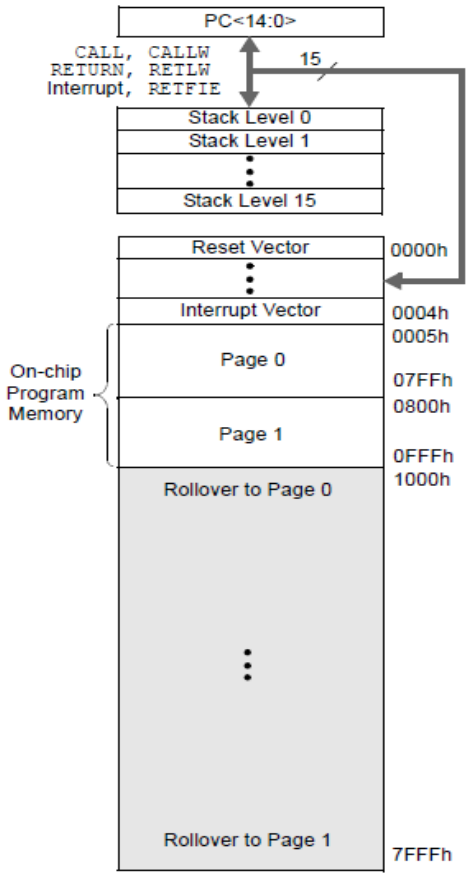


Figura 3.1: *Memoria programa*.

3.1. Definiendo características

Se requiere definir las características deseadas, JBoot está pensado para ser lo menos invasivo dentro del microcontrolador, a diferencia de otros bootloader JBoot está diseñado para ocupar la menor cantidad de memoria programa posible, no hace uso innecesario de los registros de propósito general para almacenar datos y no modificar el esquema de funcionamiento del microcontrolador, dicho de otra forma no hace re-definición de vectores.

JBoot está diseñado con el objetivo principal de solo programar el microcontrolador, al momento de escribir esta tesis no es capaz de leer la memoria programa, ni la EEPROM del microcontrolador, en futuras versiones se podría añadir estas características.

A continuación se enlistan las características que posee.

1) No re-define el vector de interrupción.

A diferencia de otros bootloader que utilizan las primeras locaciones de memoria, JBoot solo utiliza las dos primeras posiciones en memoria para apuntar al bootloader, por lo tanto el vector de interrupción se mantiene en 0x04, el único cambio que hace JBoot es sobre el vector *reset*, pasa de la dirección 0x00 a la dirección 0x02.

Esto tiene una gran importancia debido a que físicamente no se pueden re-definir los vectores, siendo la estructura de la memoria programa la mostrada en la figura anterior, por lo tanto, si el apuntador al código del bootloader ocupa o excede las primeras cuatro posiciones de memoria, es responsabilidad del programador redefinir las nuevas locaciones para los vectores, en otras palabras, un bootloader que utiliza las primeras cuatro posiciones va a ocupa más espacio porque se tiene que encargar de apuntar a los nuevos vectores, mientras que JBoot va a utilizar directamente el perfil de hardware del microcontrolador.

2) Dos modos de operación.

El bootloader debe ser capaz de saber cuándo es necesario ejecutar el algoritmo para re-programar la memoria y cuando ejecutar el programa de usuario.

Algunos bootloader utilizan un pin del microcontrolador para tal tarea, si el pin se encuentra en estado “alto” apunta al bootloader de lo contrario si el pin se encuentra en estado “bajo” apunta al programa del usuario o viceversa, la ventaja radica en que el microcontrolador no tiene que esperar ningún tiempo extra para ejecutar el programa del usuario, la desventaja a este tipo de soluciones es que se utiliza un pin extra para poder utilizar el bootloader.

JBoot define un tiempo de espera, si dentro del tiempo de espera recibe una llamada para ejecutar el algoritmo de re-programación apunta a la posición de memoria donde se aloja el bootloader, de lo contrario, si el tiempo se agota apunta a la región de memoria donde se aloja el programa del usuario, este tiempo puede ser definido para que sea lo menos invasivo posible.

3) Comunicación serial.

JBoot utiliza el módulo EUSART del microcontrolador para recibir la nueva información del firmware, se elige este módulo por la flexibilidad que presenta, por una parte en el mercado existen una gran cantidad de módulos de radio frecuencia que se comunican mediante un protocolo serial y por otra parte le permite al usuario la posibilidad de reprogramar la memoria programa, esto le permite obtener los mejor de dos mundos sin necesidad de hardware adicional.

4) Retro-inserción de apuntadores.

JBoot aprovecha las nuevas características de la serie XLP para re-programarse en memoria cada vez que entra en modo de programación de la memoria programa, esto garantiza que no importa si se interrumpe la programación de la memoria, JBoot puede volver a entrar al algoritmo y volver a re-programar toda la memoria, mientras que para series legacy los apuntadores hacia el código de JBoot deben ser especificados en el programa principal.

Cabe señalar que esto no es exclusivo para los microcontroladores de microchip, esto aplica a cualquier microcontrolador de cualquier fabricante que permita leer, escribir y borrar la memoria programa.

5) Comprobación de suma.

Por cada bloque de datos, JBoot calcula la comprobación de sumas para ser comparadas respecto a las del archivo fuente, esto lo hace antes de vaciar los datos en memoria, si por algún motivo no concuerdan, JBoot vuelve a pedir confirmación de datos, una vez que las sumas concuerdan se procede a la grabación de los datos en memoria, esto garantiza la confiabilidad a la hora de la programación en memoria programa.

6) Ocupar la menor cantidad de memoria programa para su funcionamiento.

JBoot utiliza solamente 170 locaciones de memoria programa, comparado con otros bootloader para la misma familia que ocupan alrededor de 1000 locaciones de memoria, esto representa solo el 3% de la memoria programa total del microcontrolador, esta reducción se logra gracias a que JBoot no redefine los vectores.

7) No utiliza FSR para almacenar los datos.

JBoot no hace uso de direccionamiento indirecto para almacenar los datos recibidos en los registros de propósito general, todo lo graba en los latches del microcontrolador es por tal motivo que solo los graba en memoria si la comprobación de sumas es válida, por lo tanto, en cada reprogramación de la memoria programa, no resta ciclos de escritura útil a los registros de propósito general.

3.2. Organización en memoria de JBoot

En la Figura 3.2.1 se muestra la organización en memoria de JBoot.

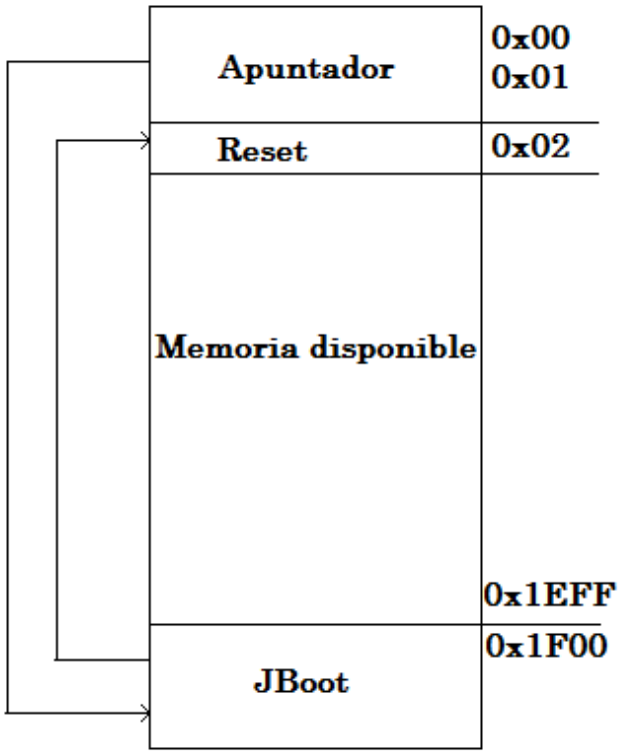


Figura 3.2.1: *Organización en memoria de JBoot.*

Una vez definidas las características el siguiente paso es programar el algoritmo a partir del diagrama de flujo mostrada en la Figura 3, la cual muestra los bloques requeridos para cada etapa.

3.3 Diagrama de flujo de JBoot:

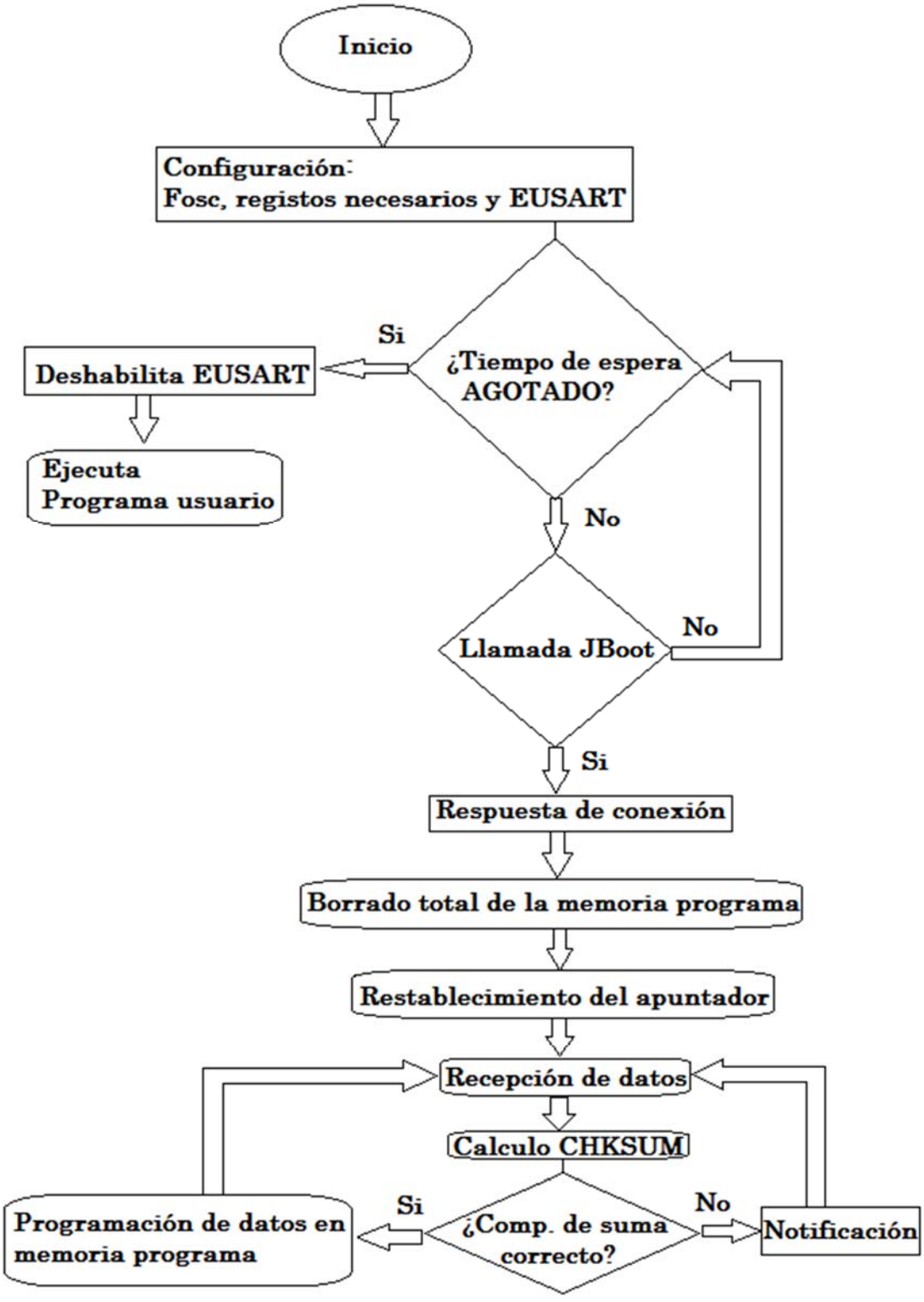


Figura 3.3: *Diagrama de flujo JBoot.*

3.4 Series XLP vs Legacy

Para la comparación se va a tomar encuentra que serie de microcontrolador soporta, la serie XLP (Extreme Low Power) fue lanzada por microchip en el año 2009, cualquier microcontrolador anterior a este año es considerada serie legacy.

Tabla comparativa de características que presenta cada serie para el diseño de un bootloader	
Seria XLP	**Serie Legacy**
• ID de revisión: 5-14 bits. • ID de dispositivo: 9- 14 bits. • Registros de configuración (Configuration Word): 2-3. • Longitud de registro de configuración: 14 bits. • Bloques de borrado: 32 palabras. • Bloques de programación: 32 palabras. • Posibilidad de borrar la memoria completa: Si. • Latches de datos: Si. • Latches de dirección: Si. • Programación de locación simple: Si • Llenado de locaciones requerida: No. • Direccionamiento indirecto: Si.	• ID de revisión: 5 bits. • ID de dispositivo: 9 bits. • Registros de configuración (Configuration Word): 2. • Longitud de registro de configuración: 14 bits. • Bloques de borrado: 16 palabras. • Bloques de programación: 4 y 8 palabras. • Posibilidad de borrar la memoria completa: No. • Latches de datos: No. • Latches de dirección: Si • Programación de locación simple: Solo en configuración de palabra. • Llenado de locaciones requerida: Si, el diseñador lo debe implementar en software. • Direccionamiento indirecto: Si.

Cuadro 3.4.1: *Diferencias entre XLP y Legacy.*

3.5 Comparación JBoot frente a otros bootloader

Se va a comparar JBoot frente a varios bootloader entre ellos el proporcionado por el fabricante del microcontrolador, existen otros bootloaders pero los diseñadores no han liberado el código para poder ser comparado, de todas maneras se incluyen en el cuadro 3.5.1.

El Cuadro 3.5.1 muestra la comparación entre diferentes bootloaders:

Nombre	**Series suportadas**	**Tamaño (Espacio)**	**FRS**	**Comunicación**
JBoot	XLP/Legacy	170	No	UART
Tiny bootloader	Legacy	100	Si	UART
Microchip	XLP/Legacy	-	Si	USB/UART/I2C
WLoader	Legacy	1000	-	UART
KarlLunt	Legacy	512	-	UART
PICLOADER	Legacy	2000	-	UART
bootload	Legacy	800	Si	UART
CodeLoader	Legacy	1000	-	UART
Jolt	Legacy	512	-	UART
SGupta	Legacy	256	-	UART
Jogu	Legacy	-	Si	UART
MikroBootloader	XLP/Legacy	-	-	USB
jackdoll	Legacy	768	Si	UART
Pickme Bootloader	Legacy	183	Si	I/O Pin

Cuadro 3.5.1: *Cuadro de comparación.*

La mayoría de los bootloader comparados utilizan la programación de los datos mediante lotes, mientras que JBoot lo hace en línea, la ventaja reside en que no se hace uso de los registros de propósito general los cuales tienen un ciclo de acceso limitado, por lo tanto no se desgasta su ciclo y no existe latencia lo que se traduce en una programación de memoria más rápida.

Por otra parte solo los bootloaders de empresas soportan la familia XLP de microchip, y teniendo en cuenta que estos bootloader generalmente están diseñados en lenguajes de programación de alto nivel el espacio que utilizan en memoria programa es mayor, esto significa que el usuario puede llegar a tener limitaciones en cuanto al tamaño de su programa principal, mientras que JBoot solo utiliza los últimos 170 locaciones de memoria baja.

Por último cabe señalar que los bootloader creados por empresas soportan diversos protocolos de programación y no solo mediante UART, aunque cabe hacer la aclaración que solo los microcontroladores de la serie 18F cuentan con un módulo USB de lo contrario sería necesario añadir el modulo al diseño, también se debe tener en cuenta que diversos módulos de radiofrecuencia utilizan comunicación serie basados en RS-232, la flexibilidad que supone utilizar un módulo EUSART es mayor.

Capítulo 4

JLoader

Un bootloader por sí mismo es inútil, a menos que el firmaware de la aplicación no sea mayor a 10 líneas de código y se esté dispuesto a programar la memoria por bloques de 8 palabras a la vez, de forma que, el usuario tenga que decodificar el archivo manualmente y luego enviarlo al bootloader, por lo tanto es necesario diseñar un servidor que sea capaz de decodificar el archivo hexadecimal para luego transmitirlo al bootloader, es por tal motivo que se decidió diseñar un cliente llamado JLoader, el cual es capaz de entender cualquier archivo hexadecimal independientemente del fabricante y del ensamblador/compilador.

El corazón del servidor es, la capacidad de entender (decodificar) el formato del archivo hexadecimal generado por el ensamblador o compilador, este archivo utiliza una codificación conocida como formato Intel.

4.1 Formato Intel

El archivo hexadecimal es una forma de representar un archivo binario en formato ASCII, debido a que está en formato ASCII en vez de binario es posible guardarlo en cualquier medio no binario como por ejemplo tarjetas perforadas, cintas magnéticas, etc.

Existen diferentes formatos, los cuales son: Intel hex 8 bits, intel hex 16 bits e intel hex 32 bits.

La versión intel hex de 8 bits permite programar dentro de los primeros 16-bits lineales de la memoria, intel hex de 16 bits permite programar bloques de 20 bits segmentados de la memoria, e intel hex 32 bits permite programar 32 bits lineales dentro de la memoria.

Actualmente existen seis tipos diferentes:

- Registro dato (formatos 8,16 y 32 bits)
- Registro final de archivo (formatos 8,16 y 32 bits)
- Registro de segmento dirección extendida (formatos 16 y 32 bits)
- Registro de inicio de segmento de dirección (formatos 16 y 32 bits)

- Registro de dirección lineal extendida (formato 32 bits)
- Registro de inicio de dirección lineal (formato 32 bits)

Registro de datos general

MARK	RECLEN	OFFSET	RECTYPE	DATA	CHKSUM
1-byte	1-byte	2-byte	1-byte	n-bytes	1-byte

MARK.- Cada uno de los registro del archivo comienza con una marca 0x3A

RECLEN.- Especifica el número de bytes de datos o información.

OFFSET.- Especifica los 16-bits de inicio de carga, por lo tanto este campo solo puede ser utilizado para datos, en registros donde este campo no es utilizado, esta codificado como 0x30303030.

RECTYPE.- Es utilizado para interpretar la información remanente dentro del registro, este registro puede ser:

- 00 Registro Datos
- 01 Fin de Archivo
- 02 Segmento de Dirección Extendida
- 03 Inicio de Segmento de Direcciones
- 04 Dirección Lineal Extendida
- 05 Inicio de Dirección Lineal

DATA.- Consiste en cero o más bytes codificados en pares de dígitos hexadecimales, la interpretación de este campo depende del valor de RECTYPE.

CHKSUM.- Cada registro termina con una suma de comprobación que contiene la representación ASCII hexadecimal en complemento a dos del resultado de la suma de la conversión de cada par de ASCII a dígitos hexadecimales.

Registro de Dirección Lineal Extendida

MARK	RECLEN	OFFSET	RECTYPE	ULBA	CHKSUM
1-byte	1-byte	2-byte	04	2-byte	1-byte

El formato de 32-bits de registro de dirección lineal extendida es utilizado para especificar la dirección lineal base o LBA por sus siglas en inglés, tiene una longitud de 32 bits, donde los bits 0-15 son cero y los bits 16-31 son definidos como la parte superior o ULBA por sus siglas en ingles.

La dirección de memoria absoluta para datos subsecuentes se obtiene mediante la siguiente ecuación:

(LBA+DRLO+DRI) MOD 32
DRLO -> Offset en el campo de dato
DRI -> Índice dentro del byte de dato
MOD 32 -> 2^32

Se omiten los acarreos para evitar un *warparound* (0xFFFFFFFF a 0x00000000).

MARK.- Contiene un valor 0x3A.

RECLEN.- Contiene una cadena ASCII 02 (0x02) la cual es la longitud en bytes de ULBA.

OFFSET.- No se utiliza, contiene una cadena ASCII 00 (0x3030).

RECTYPE.- Contiene una carácter ASCII 04 (0x04) el cual corresponde a un tipo "Dirección Lineal Extendida"

ULBA.- Especifica el valor de 16 bits de ULBA.

CHKSUM.- Contiene el resultado de la suma de los campos RECLEN, OFFSET, RECTYPE y ULBA en complemento a dos.

Registro de segmento dirección extendida

MARK	RECLEN	OFFSET	RECTYPE	USBA	CHKSUM
1-byte	1-byte	2-bytes	02	2-bytes	1-byte

El formato de 16 bits de registro de segmento de dirección extendida es utilizada para especificar los bits 4-19 del segmento de dirección base o SBA por sus siglas en inglés, donde los bits 0-3 son cero, los bits 4-19 son definidos como segmento superior de dirección base o USBA por sus siglas en inglés.

La dirección de memoria absoluta para datos subsecuentes se obtiene mediante la siguiente ecuación:

SBA + ([DRLO+DRI]) MOD16

DRLO -> Offset en el campo de dato
DRI -> Indice dentro del byte de dato
MOD16 -> 2^16

Se omiten los acarreos para evitar un *warparound* (0xFFFFFFFF a 0x00000000).

MARK.- Contiene un valor 0x3A.

RECLEN.- Contiene una cadena ASCII 02 (0x02) la cual es la longitud en bytes de USBA.
OFFSET.- No se utiliza, contiene una cadena ASCII 00 (0x3030).
RECTYPE.- Contiene una carácter ASCII 02 (0x02) el cual corresponde a un tipo "Segmento de Dirección Extendida".
USBA.- Especifica el valor de 16 bits de USBA.
CHKSUM.- Contiene el resultado de la suma de los campos RECLEN, OFFSET, RECTYPE y USBA en complemento a dos.

Registro Dato

MARK	RECLEN	OFFSET	RECTYPE	DATA	CHKSUM
1-byte	1-byte	2-byte	00	n-bytes	1-byte

El registro de dato provee un set de dígitos hexadecimales que representan el código ASCII para bytes de datos que crean una porción de la imagen de la memoria programa.

Los campos RECLEN y OFFSET están rotados dos lugares hacia la izquierda.

MARK.- Contiene un valor 0x3A.

RECLEN.- Este campo contiene dos dígitos hexadecimales ASCII que especifican el número de bytes de datos, el valor máximo es 'FF' o 0x4646.
OFFSET.- Este campo contiene cuatro dígitos hexadecimales ASCII que representan el offset de LBA o SBA definiendo la dirección en donde se va a situar el primer byte de dato.

RECTYPE.- Contiene una cadena ASCII 00 (0x3030) que corresponde a un tipo “Registro Dato”.

DATA.- Este campo contiene pares de dígitos hexadecimales ASCII, un par por cada byte de dato.

CHKSUM.- Contiene el resultado de la suma de los campos RECLEN, OFFSET, RECTYPE y DATA en complemento a dos.

Registro de inicio de dirección lineal

MARK	RECLEN	OFFSET	RECTYPE	EIP	CHKSUM
1-byte	1-byte	2-byte	05	4-byte	1-byte

El registro de inicio de dirección lineal es utilizado para especificar la dirección de ejecución de inicio, el valor dado es de 32 bits para el registro EIP, note que este registro solo especifica la dirección del código dentro los 32 bits de dirección de espacio lineal del 80386, si el código va a empezar la ejecución del 80386 en modo real, entonces el registro de inicio de segmento de dirección debe ser utilizado en vez de este debido a que el registro especifica ambos registros CS e IP necesarios para el modo real.

MARK.- Contiene un valor 0x3A.

RECLEN.- Contiene una cadena ASCII 04 (0x04) la cual es la cantidad en bytes del registro EIP dentro de este campo.

OFFSET.- No se utiliza, contiene una cadena ASCII 00 (0x3030).

RECTYPE.- Contiene una carácter ASCII 05 (0x05) el cual corresponde a un tipo “Inicio de Dirección Lineal“.

EIP.- Este campo contiene ocho ASCII dígitos hexadecimales que especifican los 32 bits.

CHKSUM.- Contiene el resultado de la suma de los campos RECLEN, OFFSET, RECTYPE y EIP en complemento a dos.

Registro de inicio de segmento de dirección

MARK	RECLEN	OFFSET	RECTYPE	CS/IP	CHKSUM
1-byte	1-byte	2-byte	03	4-bytes	1-byte

El registro de inicio de segmento de dirección es utilizado para especificar la dirección de ejecución de inicio, el valor dado es corresponde al segmento de dirección de 20 bits para los registros CS e IP, noté que este registro solo especifica la dirección del código dentro del espacio de direcciones segmentadas de 20 bits para el 8086/80186.

MARK.- Contiene un valor 0x3A.

RECLEN.- Contiene una cadena ASCII 04 (0x04) la cual es la cantidad en bytes del registro CS/IP contenido dentro de este campo.

OFFSET.- No se utiliza, contiene una cadena ASCII 00 (0x3030).

RECTYPE.- Contiene una carácter ASCII 03 (0x03) el cual corresponde a un tipo “Inicio de Segmento de Direcciones“.

CS/IP.- Este campo contiene ocho dígitos hexadecimales ASCII que especifican los 16 bits para CS y 16 bits para IP.

CHKSUM.- Contiene el resultado de la suma de los campos RECLEN, OFFSET, RECTYPE y CS/IP en complemento a dos.

Registro final de archivo

MARK	RECLEN	OFFSET	RECTYPE	CHKSUM
1-byte	1-byte	2-byte	01	FF

El registro final de archivo especifica el fin de un archivo hexadecimal.

MARK.- Contiene un valor 0x3A.

RECLEN.- Contiene una cadena ASCII 00 (0x3030) desde que este registro no contiene información ni datos tiene un valor de cero.

OFFSET.- No se utiliza, contiene una cadena ASCII 00 (0x3030)

RECTYPE.- Contiene una carácter ASCII 01 (0x01) el cual corresponde a un tipo “Fin de Archivo“.

CHKSUM.- Desde que todos los campos son estáticos su valor siempre va a tener un valor hexadecimal 0x4646 o el carácter ASCII 'FF'.

El Cuadro 4.1 muestra un resumen de todos los registros vistos hasta ahora.

Tabla de registros						
Tipo de registro	**MARK (Byte)**	**RECEL (Byte)**	**OFFSET (Byte)**	**RECTYPE (Byte)**	**ULBA (Byte)**	**CHKSUM (Byte)**
Dirección lineal extendida	1	1	2	04	2	1
Segmento dirección extendida	1	1	2	02	2	1
Inicio de dirección lineal	1	1	2	05	4	1
Inicio de segmento de dirección	1	1	2	03	4	1
Final de archivo	1	1	2	01	-	1
Dato	1	1	2	00	n	1

Cuadro 4.1.1: *Resumen de los registros.*

4.2 Funcionamiento de JLoader

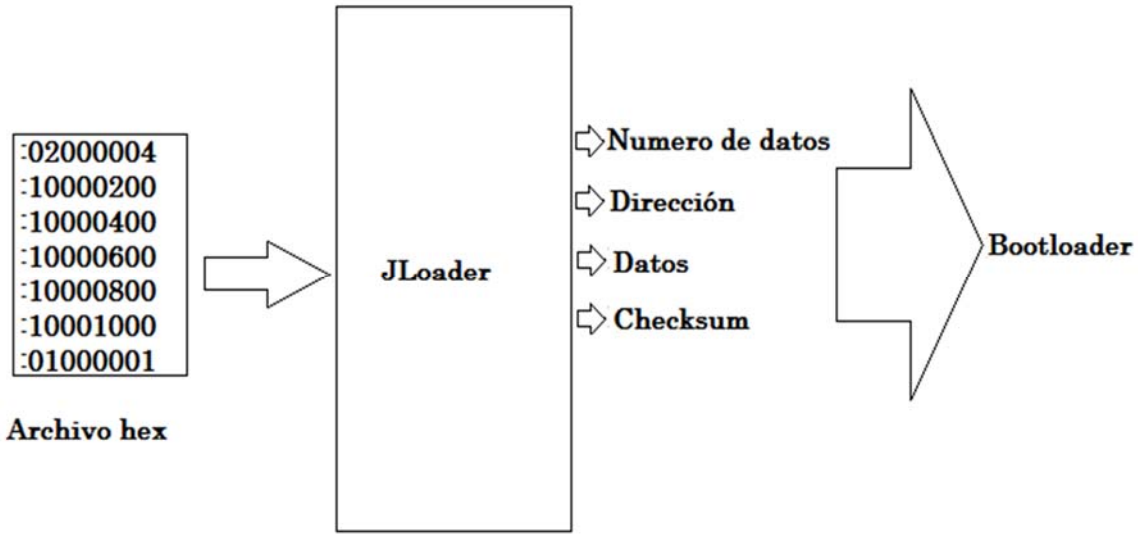


Figura 4.2.1: *Funcionamiento de JLoader.*

Lo primero que hace JLoader es buscar la marca de inicio, sin importar si se trata del formato 8,16 o 32 bits, en todos los casos el primer campo se trata de un carácter cuya longitud es de 1 byte y siempre su valor es igual a 0x3A, o lo que es lo mismo el carácter equivalente ASCII ":", una vez encontrada la omite y la cadena restante la guarda en una matriz de longitud n.

El valor de la primer casilla de la matriz corresponde al número total de datos que se van a enviar al bootloader, este campo tiene una longitud de 1-byte y nunca puede ser mayor a una cadena ASCII '10' o en formato hexadecimal 0x3031, esto significa que los datos no pueden ser mayor a 8 palabras por renglón.

El segundo valor de la matriz corresponde a la dirección destino de los datos, este campo tiene una longitud de 2-bytes por lo tato su valor máximo hexadecimal es oxFFFF, esto quiere decir que la dirección máxima a la cual se puede acceder es 0x7FFF.

El tercer valor de la matriz corresponde al tipo de formato 8,16 o 32 bits, el cual tiene una

longitud de 1 -byte, JLoader lo identifica y dependiendo del formato maneja la cadena restante de datos.

Los siguientes valores de la matriz corresponden a los nemonicos del programa principal, estos datos son enviados en paquetes cuya longitud es de 16 bits.

La última posición de la matriz corresponde a la suma de comprobación o *checksum*, la cual tiene una longitud de 1-byte.

Para asegurar que los datos se hayan enviado y recibido de forma correcta, JLoader ejecuta una doble verificación, una verificación interna y una verificación con la calculada por el bootloader.

La primer verificación consiste en comparar el cheksum calculado por el software frente el calculado por el ensamblador/compilador, esta suma se encuentra en el archivo fuente, si no coinciden las sumas es un indicativo de que hubo un error interno, ya sea por parte del ensamblador/compilador o bien por JLoader, como hubo un error es indicativo de que los datos están corrompidos, JLoader manda una señal de error al bootloader para que vuelva a cargar los datos nuevamente, se vuelve a comparar ambas sumas y en caso de que vuelva a existir un error JLoader le notifica al usuario de que existe un problema con el archivo fuente.

La segunda verificación consiste comparar el checksum calculado por JLoader frente el calculado por el bootloader, primero JLoader le solicita el checksum calculado al bootloader, una vez que es recibido hacer la comparación si son iguales le manda una señal de confirmación para que programe la memoria programa, de lo contrario, en caso de que sean diferentes le manda una señal de error y le vuelve a enviar los datos, en caso de que nuevamente la comprobación de sumas falle, JLoader le muestra una notificación de error al usuario.

- Portabilidad de JLoader:

Aunque JLoader fue creado utilizando net framework 4.5 puede ser exportado a otro lenguaje de programación dado que el corazón del programa es el traductor intel hex 32,

para hacer el *port* del cliente hacia otro lenguaje o sistema operativo solo se requiere traducir las instrucciones de un lenguaje de programación en su equivalente al lenguaje que se quiera utilizar.

JLoader nació en el año 2015 es por tal motivo que se utilizaron las librerías net frame, pero técnicamente estas librerías no influyen en el funcionamiento de JLoader y esto es debido a que no se están utilizando tecnologías que no se utilizaran ya hace años atrás, por ejemplo JLoader hace uso de comunicación serie basada en el protocolo RS-232 y no en paquetes USB por lo tanto se puede utilizar el control de comunicación de visual basic 6.0 para enviar y recibir datos, por lo tanto JLoader puede ser utilizado en sistemas de hace varios años atrás lo cual lo hace una herramienta versátil.

4.3 Requerimientos de sistema para utilizar JLoader

1. Sistema operativo: Windows 7/8/8.1/10 (32/64 bits).
2. Procesador: 1.7Ghz o superior.
3. Memoria RAM: 256Mbytes o más.
4. Espacio en disco duro: 10 Mbyte o más.
5. Versión Netframe: 4.5 o superior.
6. Puerto de comunicación: Puerto serie compatible con protocolo RS-232.

4.4 Funcionamiento JLoader-JBoot

La Figura 4.4.1 muestra el esquema de funcionamiento de JLoader y JBoot.

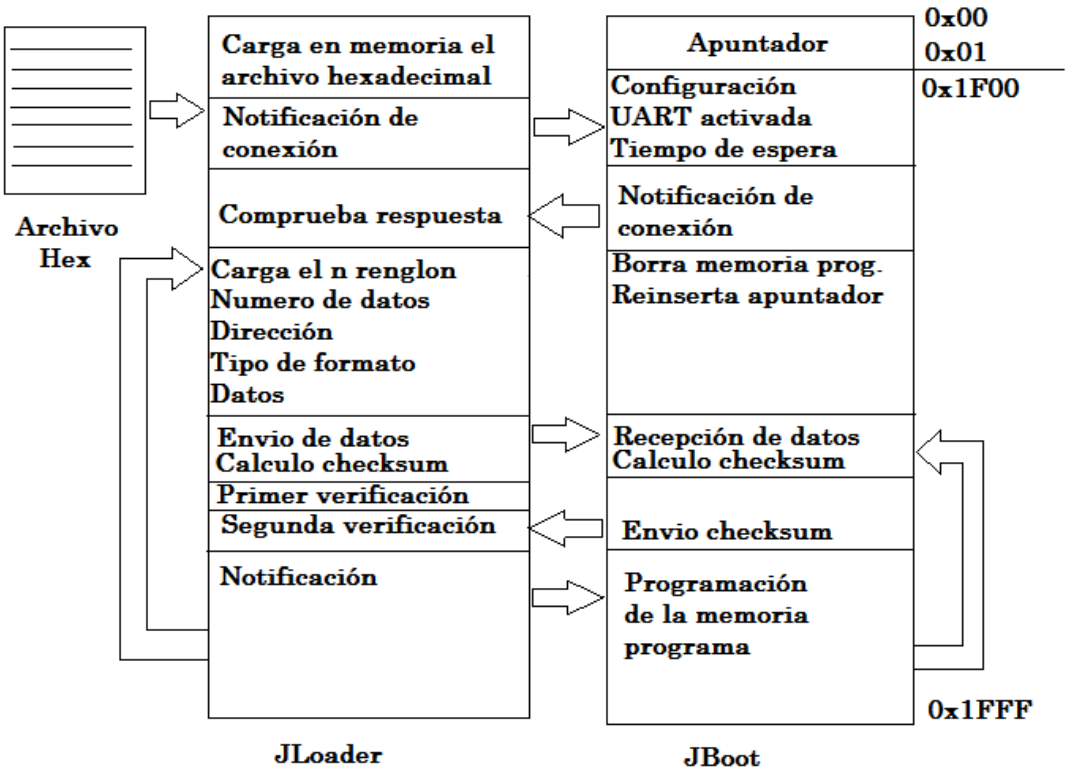


Figura 4.4.1: *Funcionamiento JLoader-JBoot.*

JLoader se comunica con el cliente y dependiendo de la respuesta JLoader responde a la solicitud, en caso de que el cliente sea basado o se trate de JBoot le manda una notificación de conexión, en caso contrario JLoader le muestra al usuario un mensaje de error, esto se debe a que tal vez el bootloader del cliente no sea compatible con JLoader, una vez establecida la conexión suceden dos cosas, por parte de JBoot borra la memoria programa y re-inserta los apuntadores en memoria, por parte de JLoader carga el archivo hexadecimal y lo decodifica, dado que al microcontrolador le toma unos pocos milisegundos en terminar las tareas se queda en espera de JLoader, una vez que JLoader está listo envía los datos, cuando se termina el bloque ambos calculan la suma de comprobación, JLoader hace dos revisiones la primera es interna, compara el checksum leído frente al calculado, si es correcto le pide a JLoader su comprobación de sumas, JLoader lo compara y si son iguales le envía la notificación para que JBoot pueda programar en memoria los datos, en caso

contrario le envía una notificación de error, JBoot no programa los datos en memoria programa, reinicia los contadores y se queda en espera de los nuevos datos.

En el siguiente capítulo se propone una aplicación para el microcontrolador PIC16F886, JBoot tiene un bootloader para este microcontrolador, el cual es J16F886.

Capítulo 5

Aplicaciones

5.1 Letreros LED

Cada vez es más común ver anuncios que utilizan tecnología led, esto es, pantallas que están formadas por matrices de led, estos anuncios tienen como objetivo dar a conocer desde productos hasta el estado de la bolsa de valores mundial.

Para mostrar esta información contamos con diferentes tipos de pantallas, las cuales pueden dividirse en dos grupos:

El primer grupo corresponde a las matrices que no tienen incorporadas funciones de control por lo tanto, el usuario es el responsable de programar las secuencias necesarias para mostrar el texto, y el segundo para aquellas que traen integrado las funciones de control, por lo tanto solo es necesario programarlas para mostrar el texto deseado.

Las ventajas que presenta una frente a la otra depende exclusivamente de la aplicación, en términos generales, las pantallas con funciones de control integradas son utilizadas únicamente para mostrar texto, no tienen la posibilidad de personalización del tipo de fuente ni del tamaño, su capacidad de animación es mínima, la ventaja que presenta es que no es necesario programar las secuencias para mostrar el texto.

Por otra parte las pantallas sin funciones de control presentan características avanzadas de animación y de personalización, la desventaja es que es necesario diseñar las secuencias de control.

5.1.1 Planteamiento del problema

El Gobierno de la Ciudad de México ésta organizando una exposición de proyectos de diferentes universidades, entre ellas se encuentra la Universidad Autónoma de la Ciudad de México, el único requerimiento que se exige es que cada estand muestre las siglas de la institución a la cual pertenecen los proyectos.

Este cartel puede ser hecho en papel o bien mediante el uso de pantallas led.

Los requisitos en caso de utilizar una pantalla led son:

- Las letras pueden estar en mayúsculas o minúsculas.
- No hay restricción en cuanto al color del led.
- El tamaño de la pantalla deber ser no mayor de 12mm x 12mm esto por cuestiones de espacio que se asigna a cada universidad la cual consta de una mesa de 120.4 x 75 cm para mostrar los proyectos.

5.1.2 Desarrollo

Para mostrar texto sobre una pantalla, la mejor solución es utilizar una pantalla con funciones de control integradas.

Como se mencionó en un principio, una de las ventajas que provee las pantallas con funciones de control es que no se necesitan hacer los algoritmos de control para mostrar texto, por lo tanto, solo es necesario programar el orden en el cual se van a ir mostrando las letras.

Para este ejemplo se utilizará el bootloader propuesto para demostrar cómo se agiliza el tiempo de desarrollador y las ventajas que puede presentar frente a un firmware estático.

Cabe aclarar que la utilización de un bootloader no es la única solución implementable puesto que se puede optar por algoritmos para el manejo del letrero, pero su ventaja reside en correcciones de errores o añadir nuevas funciones después de producción.

Debido a que existen restricciones en el tamaño de la pantalla se eligió la pantalla cuyo modelo es OSRAM PD4437, esta pantalla cubre todas las restricciones para diseñar el “cartel”.

Las características técnicas completas de la pantalla se encuentran en el apéndice C1.3.

Hardware:

1. Pantalla Osram PD4437.
2. PIC16F886.
3. Resistencia 10kΩ @ 1/4 watt.
4. Push botón de 2 o 4 terminales.
5. Alambre calibre 24-22AWG.

Una de las ventajas que presenta esta pantalla es que no requiere hardware adicional para controlar el nivel de brillo de los leds o módulos de comunicación especiales como podría ser el caso de pantallas con interfaz SPI u I2C.

Cuenta con un controlador interno el cual se encarga de decodificar las instrucciones que le envía el microcontrolador, por lo tanto no es necesario acceder a ningún registro interno, por otra parte tampoco es necesario codificar los caracteres, la pantalla cuenta con un generador de caracteres.

La comunicación con este dispositivo se hace mediante dos señales principales, estas señales se conectan a los pines RD y WR (ambos activos en bajo), el pin RD sirve para leer la memoria interna del controlador, mientras que el pin WR se utiliza para enviarle el carácter que se desea mostrar en pantalla, siendo los pines A0, A1, y A2 los que determinan la posición del carácter, en la Figura 5 se muestra los pines de la pantalla.

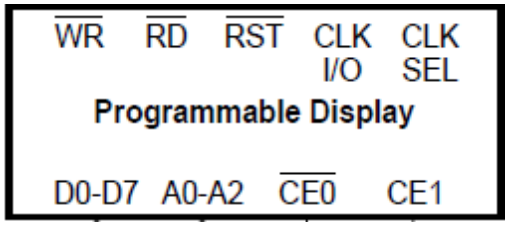


Figura 5.1.2.1: *Esquemático de la pantalla.*

Antes de poder escribir o leer la memoria de la pantalla es necesario configurarla, la secuencia de configuración consta de una serie de pasos que se deben seguir en orden, a continuación se muestra esta secuencia y un ejemplo de programación.

El formato general de la palabra de control se muestra en la siguiente figura:

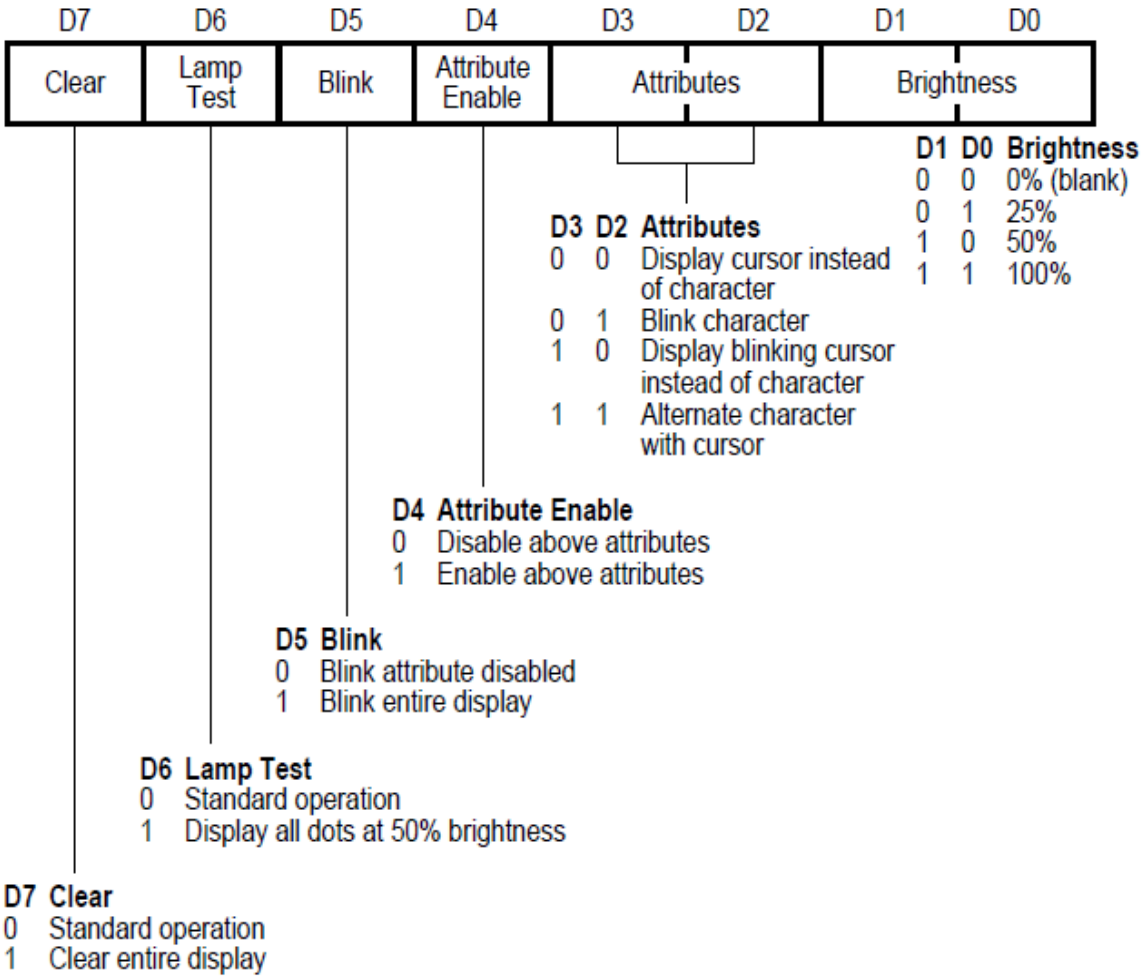


Figura 5.1.2.2: *Formato general de la palabra de control.*

Ejemplo de comandos de datos de entrada:

CE0!	CE1	RD!	WR!	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0	Operación
1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	Sin cambio
0	1	0	1	1	0	0	X	X	X	X	X	X	X	X	Lee dígito 0
0	1	1	0	1	0	0	0	0	1	0	0	1	0	0	Escribe dígito 0
0	1	1	0	1	0	1	0	1	0	1	0	1	1	1	Escribe dígito 1
0	1	1	0	1	1	0	0	1	1	0	0	1	1	0	Escribe dígito 2
0	1	1	0	1	1	1	0	0	1	1	0	0	1	1	Escribe dígito 3
0	1	1	0	1	0	0	1	X	X	X	X	X	X	X	Escribe dígito con cursor

Dado que la pantalla internamente contiene todo lo necesario para controlar las matrices led no se requiere de módulos adicionales como por ejemplo convertidores analógico digital, comparadores o módulos de comunicación, por lo tanto los requerimientos mínimos necesarios para establecer el microcontrolador serían:

1. Bus de 8 bits para datos.
2. Bus de 3 bits para dirección.
3. Bus de 2 bits para control escritura/lectura.
4. Bus de 3 bits control de activación/reset.
5. Memoria programa mayor o igual a 512 bytes.
6. Registros de propósito general mayor o igual a 256 bytes.

Para controlar la pantalla se eligió el microcontrolador pic16f886, este microcontrolador cumple con los requerimientos mínimos necesarios para controlar la pantalla, por otra parte este microcontrolador se puede localizar fácilmente en las tiendas de electrónica.

Las características técnicas del microcontrolador se encuentran en el apéndice C1.4.

La programación del código se va a dividir en tres bloques principales, el primer bloque corresponde a la configuración del dispositivo y pantalla, el segundo se encargar de

almacenar el texto y el tercer bloque se encargar de ordenar y mostrar en pantalla los caracteres.

Observaciones sobre el tamaño de memoria disponible:

En la memoria programa se define el texto a ser mostrado, el programa deja los primeros 256 bloques de memoria libres para añadir futuras características al firmware, el número total de caracteres que se pueden almacenar en memoria programa incluyendo el espacio que utiliza JBoot es de 960 caracteres, si se desea hacerlo mediante direccionamiento indirecto se debe tomar en cuenta la cantidad de memoria dato del microcontrolador, para el PIC16F886 se puede alcanzar una tamaño de almacenamiento máximo de 368 caracteres en memoria RAM.

Observaciones sobre el número de caracteres:

La variable cuya locación se encuentra en la posición 0x23 es la encargada de establecer el número total de caracteres, en este caso el registro es de 8 bits por lo tanto se pueden contabilizar un máximo de 256 caracteres por barrido, si se deseara desplegar todos los carácter en un solo barrido se puede utilizar una concatenación con otro registro para tener un tamaño 65536 caracteres.

5.1.3 Diagrama de flujo para pantalla sin animación

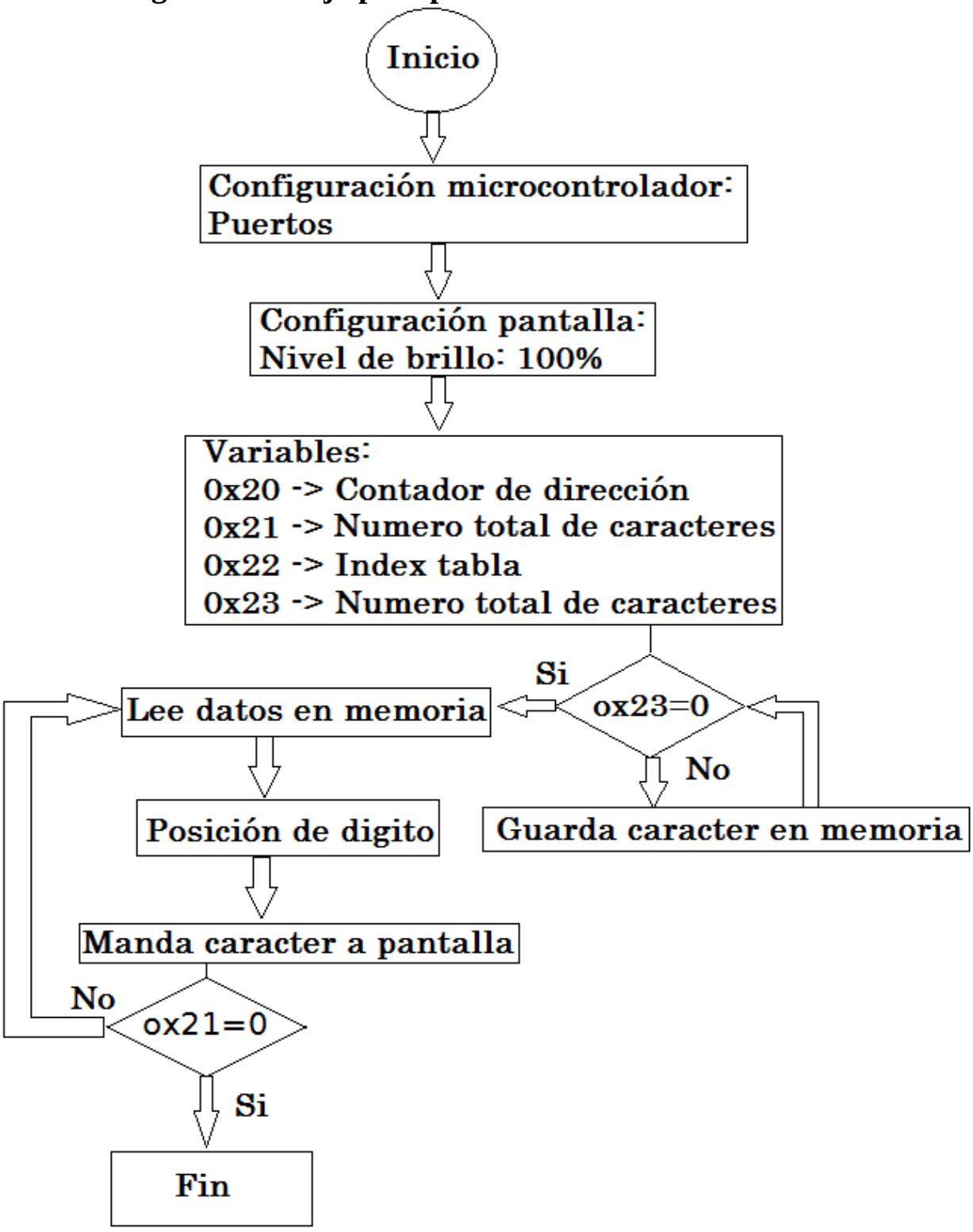


Figura 5.1.3.1: *Diagrama de flujo para pantalla sin animación.*

La estructura utilizada para mostrar el texto es muy versátil, presenta la ventaja de programación por módulos, lo que significa un firmware, al cual se le puede añadir nuevas funciones sin necesidad de modificar la estructura base.

Ahora supongamos que el prototipo ha sido aprobado y se están fabricando los "carteles" para su utilización, pero resulta que otra universidad está mostrando sus iniciales con un grado de presentación más sofisticado que simplemente texto plano.

Aquí es precisamente la ventaja de utilizar un bootloader, si solamente se hubiera programado la secuencia para mostrar la iniciales, no habría manera fácil de modificar el firmware, a menos que se hayan puestos los conectores para utilizar ICSP en el PCB, pero aun así el ICSP utiliza los pines donde va el bus de datos, en el mejor de los casos esto no tiene ningún efecto secundario, con el bootloader esto no sucede, puesto que utiliza los pines asignados al módulo para comunicación serie (EUSART), por otra parte la programación mediante ICSP requiere de un programador especializado, el cual no lo venden en todas partes, mientras que JBoot solo requiere un cable serial USB-UART o bien un convertidor de nivel COM-UART los cuales están disponible en prácticamente cualquier tienda de electrónica.

Regresando al problema inicial donde se requiere que las letras tengan cierto grado de animación, la forma más rápida y eficiente para lograr es utilizando un contador.

Obviamente se puede optar por métodos más sofisticados de presentación pero para dejar claro al lector de lo fácil y rápido que se puede añadir características a un firmware bien diseñado se optó por una animación que despliegue una letra a la vez.

El diagrama de flujo quedaría de la siguiente manera. Si se compara con el anterior, se puede observar que solo se añadió una nueva instrucción a la configuración y un bloque llamado *timer*.

La función de este bloque no es otra más que añadir un retardo para mostrar el siguiente carácter, si bien es cierto que se podría optar por un código en vez de utilizar el timer del controlador, la utilización del mismo proporciona ciertas ventajas a la hora de ejecución.

Entre una de las ventajas a destacar se encuentra que no es necesario el poleo de la bandera o tiempos muertos de procesamiento, si se compara la utilización del timer frente a un segmento de retardo, el timer presenta la ventaja de poder ser invocado mediante interrupción, por otra parte se puede escalar el tiempo del retardo de manera simple, solo bastaría cambiar el prescalador o bien ajustar los ciclos de repetición, mientras que con un retardo basado en segmento de código será necesario re calcular todo el algoritmo, por último precisión en los tiempo, mediante la utilización del módulo TMR del microcontrolador es posible lograr tiempos de retardo precisos a diferencia del código de retardo los cuales dependen en gran medida el tiempo de ejecución del código y de los ciclos que le toma a cada instrucción para lograr obtener resultados precisos cosa que en la práctica resulta ineficiente.

5.1.4 Diagrama de flujo de pantalla con animación.

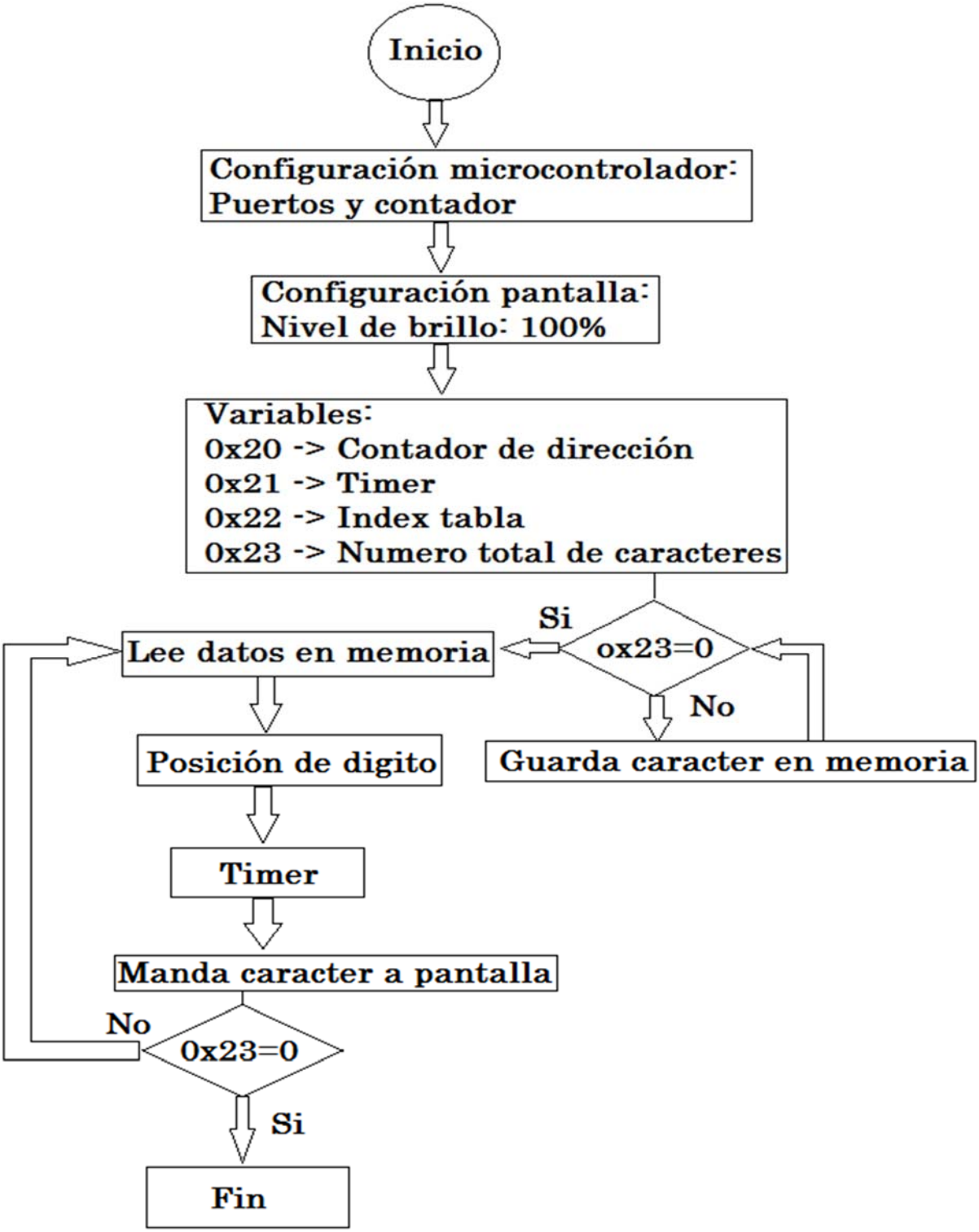


Figura 5.1.4.1: *Diagrama de Flujo de pantalla con animación.*

5.1.5 Resultados

En los siguientes enlaces se muestra el video de ambos casos

Texto sin animación:
https://www.youtube.com/watch?v=CrEUIlb5Bm8

Texto con animación
https://www.youtube.com/watch?v=wT6zQm9l5Ew

5.1.6 Conclusión

Gracias a la modularidad del firmware y a la utilización de un bootloader se pudo añadir nuevas funciones/características a nuestro diseño si necesidad de modificar el hardware, esto representa una fuerte ventaja frente a cualquier sistema pre-programado sin posibilidad de actualización.

Claramente se puede ver las ventajas que se obtienen con la utilización de un bootloader para un caso práctico real, resultó ser una solución de entre las opciones actuales la mejor, los tiempos de diseño y desarrollo fueron reducidos considerablemente.

En el siguiente capítulo se propone una metodología para poder implantar controladores digitales mediante la utilización de una metodología que se divide en tres partes la primera consiste en encontrar el modelo de la planta a controlar, la segunda parte se centra en el modelado del sistema mediante la utilización de herramientas tales como Matlab y Simulink para encontrar los valores de las ganancias que conforman el algoritmo PID, una vez encontrado todos los datos el último paso es la utilización de un microcontrolador específico para control digital.

Capítulo 6

Diseño controlador digital

Hoy en día es común ver brazos robóticos en las fábricas, la intención de estos brazos es automatizar el proceso de fabricación de ciertos productos, por ejemplo estos brazos robóticos son utilizados en las líneas de producción de un automóvil o bien para el ensamblaje de una pantalla led, pero no son exclusivos de la industria, también se les puede ver en los laboratorios de las universidades o inclusive en tiendas de electrónica como juguetes didácticos.

Sin importar el ramo en el que se encuentren, los brazos robóticos deben garantizar precisión y confiabilidad en su operación, esto se logra mediante algoritmos de control, estos algoritmos garantizan que sin importar las perturbaciones externas el brazo va a ser capaz de corregir su comportamiento para alcanzar el objetivo inicial sea este: posición, velocidad, potencia, etc.

El 90% de los controladores industriales pertenecen a la familia PID, una de las razones de su utilización es por su simpleza, no requiere necesariamente el modelo de la planta, en la mayoría de los casos se puede utilizar la estructura clásica, una de las ventajas que tiene este tipo de controlador es que solo es necesario hacer el ajuste de los tres parámetros, ahora bien, una de las desventajas es que no es muy usada en la industrial.

Existen diferentes técnicas de sintonización, una de las más utilizadas es la de Ziegler–Nichols la cual utiliza la respuesta al escalón del sistema en lazo abierto para definir las ganancias de control a partir de una tabla de valores definidos, otra técnica utilizada está basada en prueba y error para encontrar los valores de los parámetros, aquí no se cuenta con ninguna tabla de valores por lo tanto es una técnica meramente empírica.

Los métodos anteriormente expuestos presentan la ventaja de que pueden ser utilizado cuando no se cuenta con el modelo de la planta, lo cual sucede la mayor parte del tiempo (excepto en la industria), por lo tanto pareciera que estas técnicas pueden abarcar el 100% de las situaciones, pero la realidad es que no siempre se pueden utilizar o bien no es óptimo ni eficaz su utilización.

Por tal motivo se desarrolló una metodología para encontrar el modelo de la planta posteriormente se controla mediante técnicas de control. Para demostrar la metodología se va a utilizar un motor de corriente directa como planta a controlar. Una vez que se tenga el modelo se procederá a encontrar las ganancias de los parámetros del sistema de control basado en PID.

6.1 Tarjeta DAQ

Para poder encontrar la función de transferencia del motor es necesario contar con un dispositivo que sea capaz de convertir los niveles de voltaje analógicos a digitales, esto con el fin de enviar esta información a un programa de computadora, el cual sea capaz de mostrarnos mediante gráficas el comportamiento del motor a modelar, para tal efecto se utiliza una tarjeta de adquisición de datos o DAQ por sus siglas en ingles.

6.2 Serie USB-600X de NI:

En la siguiente tabla se muestran las características básicas de las tarjetas de adquisición de datos de National Instrument, se trata de la serie 6008 y 6009, estas tarjetas se van a tomar como referencia para obtener la ficha técnica y a partir de ellas proponer las características del DAQ a construir.

Cabe aclarar que se eligieron las series 6008 y 6009 debido a que son las series básicas que ofrece National Instruments. Dicho de otra forma, son las tarjetas de menor costo que se pueden adquirir.

Producto	Entradas analógicas	Resolución de entrada	Máxima taza de muestreo	Salidas analógicas	Resolución de salida	Taza de salida	Digital I/O	Contador 32bits	Trigger
USB-6008	8	12	10(kS/s)	2	12	150Hz	12	1	Digital
USB-6009	8	14	48(kS/s)	2	12	150Hz	12	1	Digital

Por parte de la cantidad de entradas analógicas solo se requiere una, por lo tanto eso no es un factor importante, la resolución en la mayoría de los microcontroladores que se pueden conseguir en México son de 10 bits, existen versiones de 12 bits en la serie 18F pero como se trata de utilizar lo que se puede encontrar se optó por un ADC de 10 bits, la taza de

muestreo es de 1.5 KHz, la resolución de salida no aplica dado que no se va a sacar los datos por un DAC, si no directamente por el puerto de comunicación serie, la taza de salida es desconocida por lo tanto se va a utilizar una velocidad de transmisión de 19200 bps, contador de 16 bits, y dos canales de salida digitales.

El Cuadro 6.2.1 presenta la tabla comparativa, entre el DAQ y las tarjetas de adquisición de datos de *National Instrument*:

Producto	**Entradas analógicas**	**Resolución de entrada**	**Máxima taza de muestreo**	**Taza de salida**	**Digital I/O**	**Contador**	**Costo (Pesos mexicanos)**
DAQ	1	10	1.5(kS/s)		2	16 bits	178.85
USB-6008	8	12	10(kS/s)	150Hz	12	32 bits	Desde 4,950
USB-6009	8	14	48(kS/s)	150Hz	12	32 bits	Desde 7,335

Cuadro 6.2.1: Comparativa DAQ vs Seie USB-600X de National Instrument.

En la tabla se puede observar que, técnicamente las tarjetas de National Instrument cuentan con mayores prestaciones, pero es difícil hacer un análisis completo dado que no se muestra las frecuencias de reloj ni el microcontrolador utilizado en dichas tarjetas.

Por parte del DAQ la frecuencia de reloj es a 8MHz utilizando el oscilador interno, el voltaje de operación es de 4.9V @ 24 grados centígrados, por lo tanto se tiene un drift en el oscilador no mayor a +2,-2%, esto significa que no es necesario la utilización de un oscilador externo, el microcontrolador en operación tiene un consumo nominal de 2mA y .01mA en reposo.

Se podría llevar el oscilador interno a su máximo de 32Mhz pero el gasto de energía para las pocas mejorar en las prestaciones no es conveniente dado que el ADC es de 10bits y no cuenta con canales DMA, lo mejor en estos casos sería optar por un microcontrolador más sofisticado, aun así, dado que solo se necesita adquirir los datos de una señal, el pic16f1827 es capaz de cumplir con la tarea, por lo tanto todo el peso recae sobre el diseño y la optimización del firmware que va a controlar la DAQ.

Este DAQ se construirá con base en el bootloader JBoot para la familia 16f182X, esto con el fin de agilizar las modificaciones que pudieran surgir a los largo de su desarrollo.

En el Cuadro 6.2.2 se resumen las características técnicas finales del DAQ:

Características técnicas del DAQ	
Microcontrolador	Pic16f1827
Canales de entrada	Un canal analógico.
Resolución del canal analógico	10 bits
Canales de salida	Dos canales digitales conectados a un puente H L298
Comunicación	Protocolo serie asíncrono 19200:8:1:N

Cuadro 6.2.2: *Características técnicas del DAQ.*

Una vez definidas las características técnicas de la tarjeta de adquisición de datos, el primer paso fue el diseño del hardware, el esquemático se encuentra en el Apéndice D1.2 para su consulta, para posteriormente programar el firmaware que va a controlar el dispositivo.

La Figura 6.2.1 muestra el DAQ diseñado.

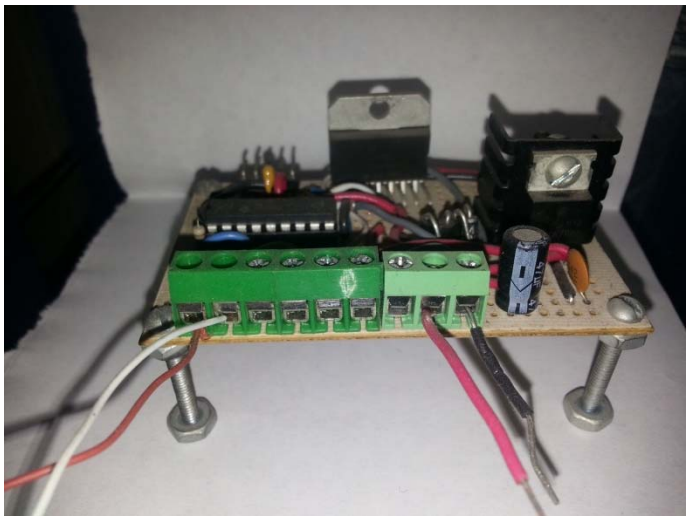


Figura 6.2.1: *DAQ*

6.3 Comparación salida del DAQ vs Osciloscopio

Para comprobar los datos que son recibidos del DAQ se acopló un engrane a la flecha del motor que a su vez mueve un engrane acoplado al potenciómetro, la función del potenciómetro es registrar los diferentes valores de voltaje que genera motor cuando este se encuentra en movimiento, es decir la posición, esta señal es adquirida por del DAQ, el cual mediante su convertidor analógico-digital registra los niveles de voltajes analógicos para posteriormente enviarlos a una computadora mediante comunicación serie, con este arreglo se pudo comparar la salida del DAQ frente a la salida de un osciloscopio.

El modelo del osciloscopio utilizado para esta comparación es Agilent Technologies DSO032A, este osciloscopio cuenta con la capacidad de exportar los datos a una memoria USB externa, lo cual resulta muy práctico para poder hacer las comparaciones, esto se debe a que entre uno de los diferentes formatos en los que puede exportar los datos el osciloscopio se encuentra el formato cvs, el cual puede ser exportado a matlab para su posterior análisis.

Por el contrario, si el osciloscopio no contara con una forma de exportar los datos entonces sería necesario utilizar los cursores del osciloscopio para moverse sobre los ejes “X” e “Y” y anotar los valores correspondientes a cada punto que conforma la señal, esta técnica en la práctica resulta viable para ser utilizada independientemente del tipo de señal que se trate, o bien, si se tienen la información suficiente sobre la señal se podría recurrir a software matemático para hacer la gráfica de la señal y posteriormente exportarlo para su análisis.

La Figura 6.3.1 se muestra el diagrama de conexión del sistema

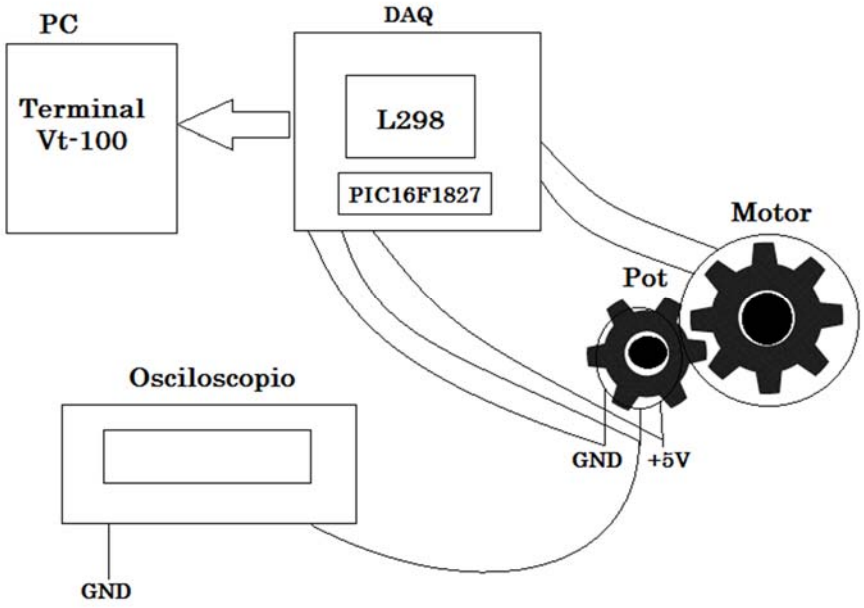


Figura 6.3.1: *Diagrama de conexión.*

El osciloscopio va a graficar el voltaje en el potenciómetro, mientras que el DAQ va a mandar los datos de convertidor ADC a una terminal Vt-100 para posteriormente graficarlos y compararlos con los datos del osciloscopio, se programó al DAQ una secuencia de movimiento, la cual consiste en llevar el potenciómetro de un extremo al otro, primero al extremo derecho y después al extremo izquierdo, dicho de otra manera, ir de 0V a 5V y después de 5V a 0V.

El microcontrolador se va a encargar de la comunicación serie y de las secuencias de giro, mientras que el puente H se va a encargar de darle la potencia necesaria para mover la flecha del motor, cabe señalar que el hardware ha sido diseñado pensando en que se está utilizando una carga reactiva, por lo tanto se añadió un sistema de protección básico basado en diodos.

6.4 Añadiendo ruido a la salida del DAQ

Cabe aclarar que el ruido inducido por el motor no se ve reflejado en la señal de salida del DAQ, esto se debe a que se está utilizando un convertidor analógico digital, la señal de

entrada se cuantifica, por lo tanto la señal resultante tiende a omitir el ruido del motor, es por tal motivo que fue necesario añadir ruido digital al firmware del DAQ.

La razón de añadirle ruido es la siguiente:

Obtener una señal lo más parecida posible con el dispositivo físico a modelar, teniendo en mente que estos datos van a ser utilizados por el software de identificación de MATLAB se espera que el sistema tenga un porcentaje de identificación mayor.

La Figura 6.4.1 muestra la comparación entre una señal sin ruido digital y una con ruido digital.

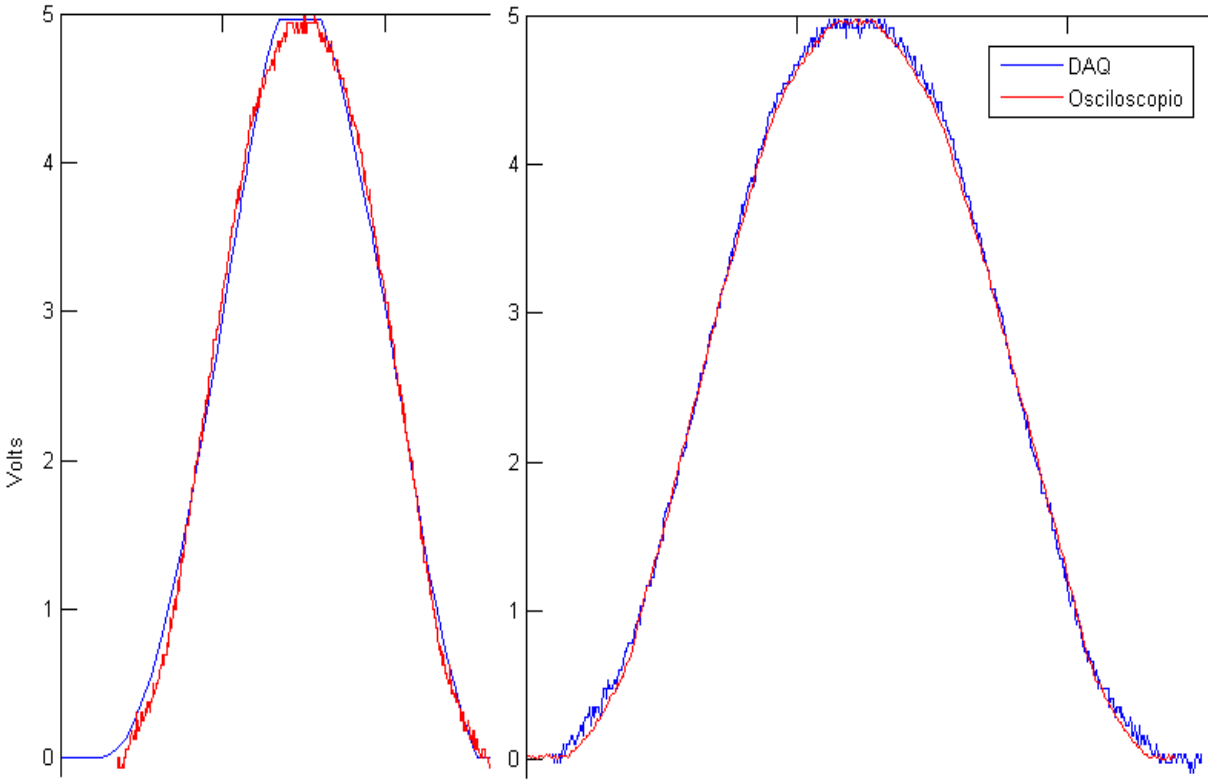


Figura 6.4.1: *Comparación señal sin ruido digital vs señal con ruido digital.*

La gráfica color rojo corresponde a la obtenida mediante el osciloscopio, mientras que la gráfica de color azul corresponde a la obtenida por el DAQ, en la gráfica de la izquierda se muestra la salida del DAQ sin ruido digital, se puede apreciar como solo se corresponden en la subida y la bajada de la señal mientras que en la curva la respuesta del DAQ es más lineal y mantiene constante la señal hasta que decrece, la gráfica de la derecha muestra la

salida del DAQ con ruido digital, se puede observar cómo se corresponde mejor con la señal del osciloscopio.

6.5 Observaciones sobre el firmware del DAQ

Es importante tomar en cuenta que muchas veces el tiempo que le toma al motor dar un giro en una dirección respecto a la otra no es el mismo, por tal motivo se decidió utilizar contadores para establecer los tiempos de secuencia.

La ventaja de utilizar contadores es que se puede modificar el número de cuentas necesarias para cada secuencia, la desventaja que esto representa es que para cada motor que se quiera modelar va a ser necesario ajustar el valor de estos parámetros.

Por otra parte la adición del ruido puede ser mediante hardware o bien mediante software, para el caso del DAQ se optó por implementarlo mediante software, mediante la utilización de rotaciones y operaciones XOR se consiguieron los números pseudoaleatorios que son sumados a la salida.

Una forma de evitar gastar tantos ciclos de reloj para la generación del ruido, puede ser implementado mediante hardware, para ello solo sería necesario una resistencia y un capacitor en paralelo, el funcionamiento de este circuito es muy simple, se trata de monitorear la carga y descarga del capacitor mediante un canal del convertidor analógico digital, los datos que se recojan se multiplican por un factor de escalamiento para posteriormente sumarlos a la salida.

La desventaja del hardware frente a la solución del software, es que se necesita calcular el tamaño del capacitor y la resistencia para obtener el factor de escalamiento, mientras que en software solo es necesario establecer el factor de escalamiento deseado.

A continuación se muestra el diagrama de flujo del firmware utilizado.

Diagrama de flujo del firmware utilizado para la comparación

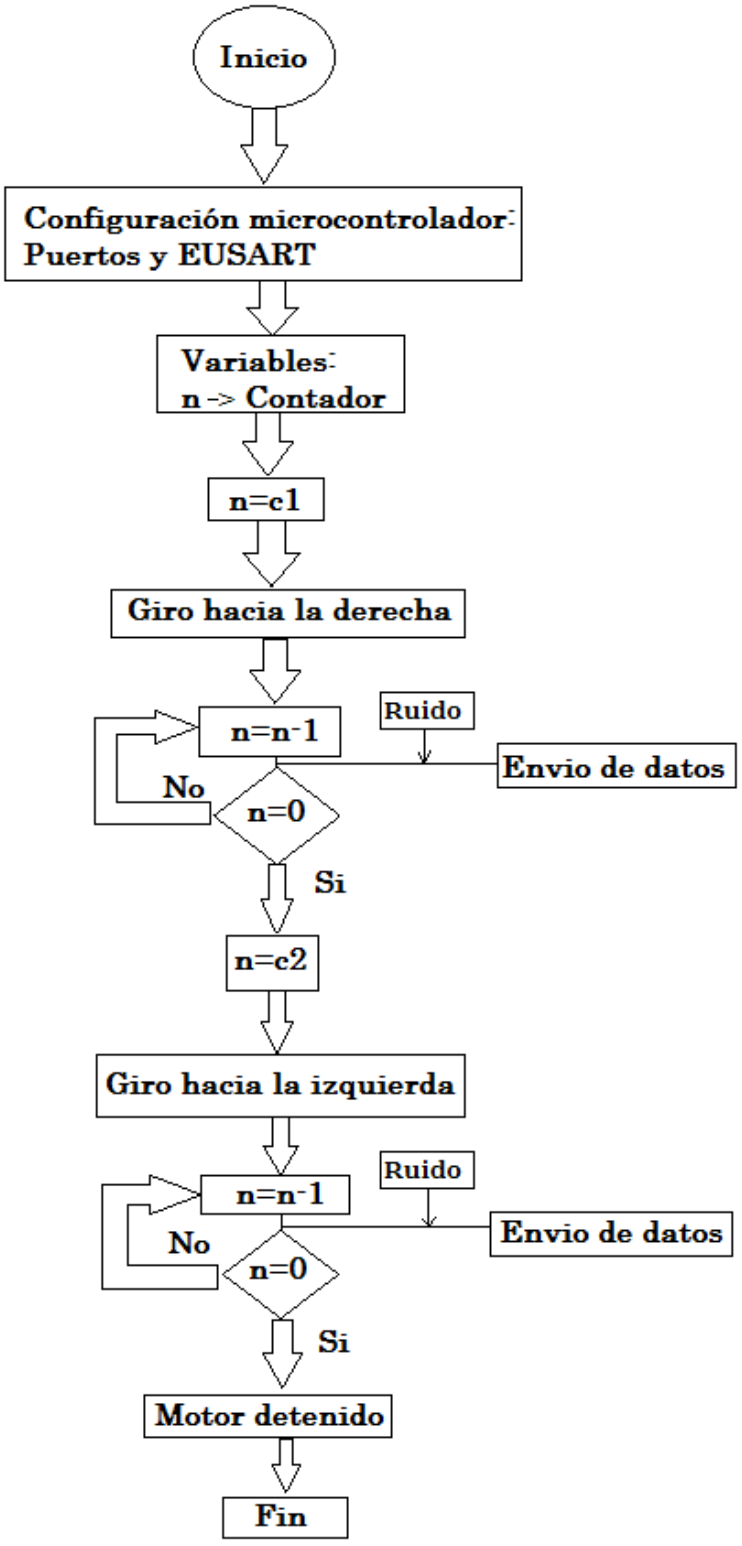


Figura 6.5.1:*Diagrama de flujo de firmware para comparación.*

6.6 Resultados experimentales

Empleando el programa MATLAB versión 2013b para graficar las salidas tanto del osciloscopio como del DAQ, se obtiene la gráfica comparativa de la Figura 6.6.1

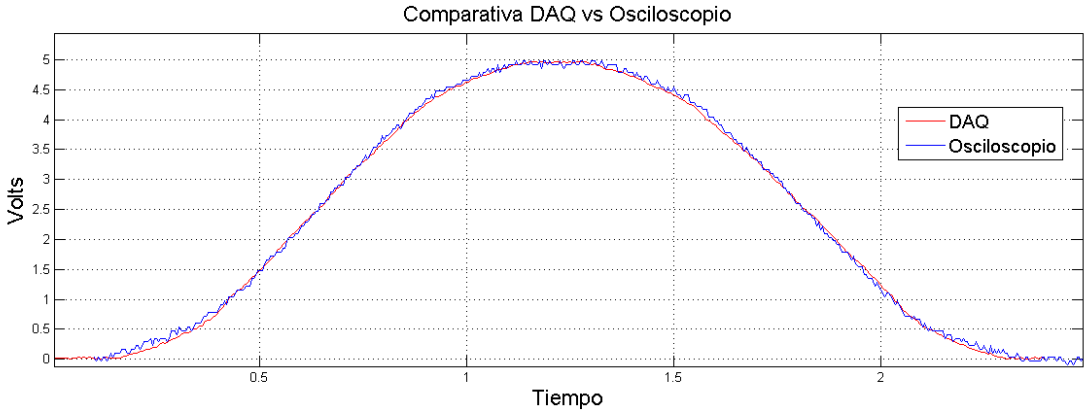


Figura 6.6.1: *Comparación entre ambas señales.*

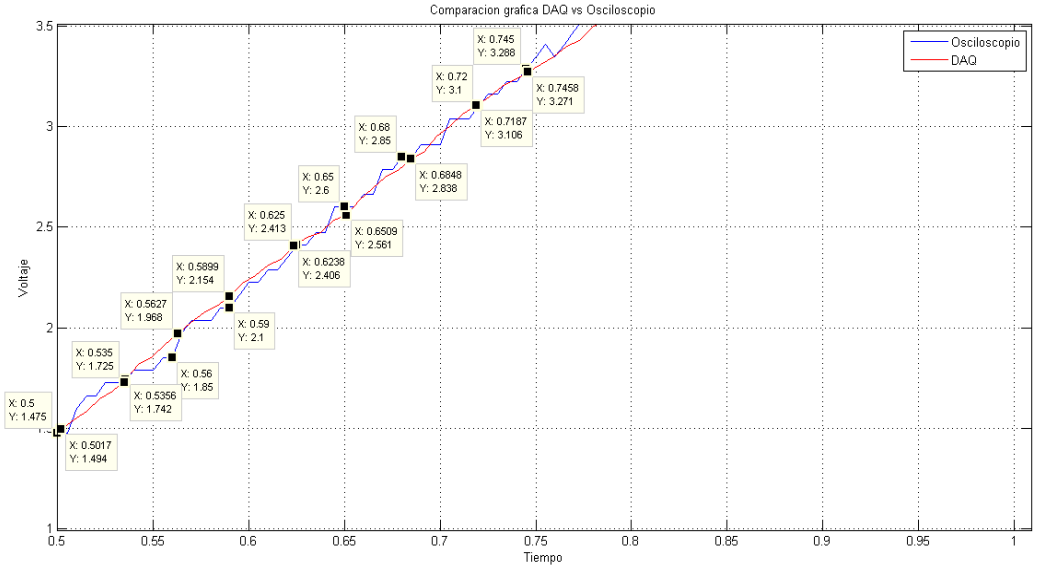


Figura 6.6.2: *Comparación de subida.*

Tabla de datos

Tiempo DAQ	Tiempo Osciloscopio	Diferencia (ms)	Valor DAQ	Valor Osciloscopio	Error (mV)
0.5017	0.5	1.7	1.494	1.475	19
0.5356	0.535	0.6	1.742	1.725	17
0.56	0.5627	-2.7	1.85	1.968	-118
0.59	0.5899	0.1	2.1	2.154	-54
0.6238	0.625	-1.2	2.406	2.413	-7
0.6509	0.65	.9	2.561	2.6	-39
0.6848	0.68	4.8	2.838	2.85	-12
0.7187	0.72	-1.3	3.106	3.1	6
0.7458	0.745	.8	3.271	3.288	-17

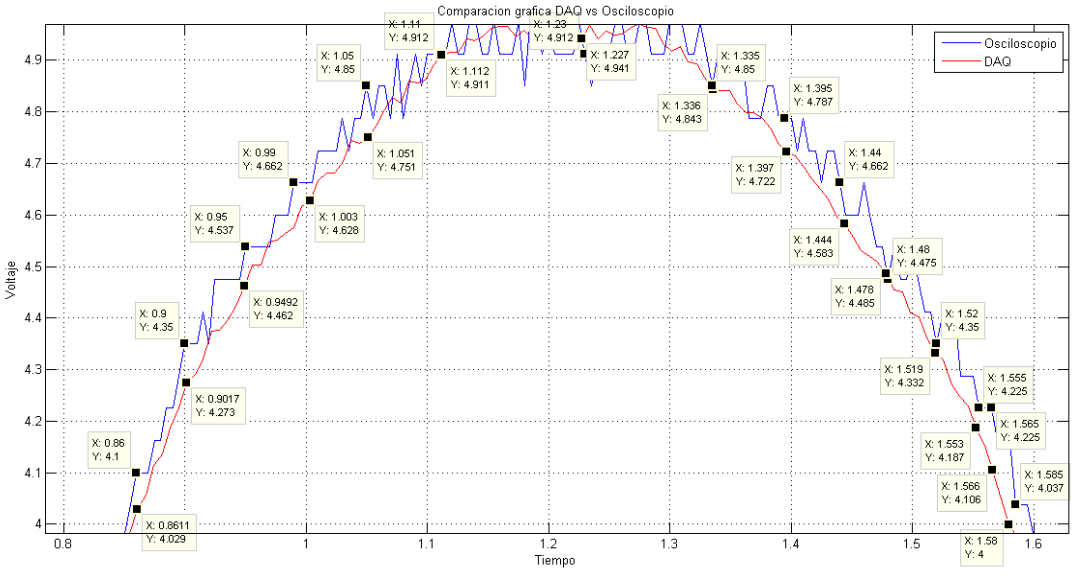


Figura 6.6.3: *Comparación cambio de sentido de giro.*

Tabla de datos:

Tiempo DAQ	Tiempo Osciloscopio	Diferencia (ms)	Valor DAQ	Valor Osciloscopio	Error (mV)
0.8611	0.86	1.1	4.029	4.1	-71
0.9017	0.9	1.7	4.273	4.35	-77
0.9492	0.95	-0.8	4.462	4.537	-75
1.003	0.99	13	4.628	4.662	-34
1.051	1.05	1	4.751	4.85	-99
1.112	1.11	2	4.911	4.912	-1
1.227	1.23	-3	4.941	4.912	29
1.336	1.335	1	4.843	4.85	-7
1.397	1.395	2	4.722	4.787	-65
1.444	1.44	4	4.583	4.662	-79
1.478	1.48	-2	4.485	4.475	10
1.519	1.52	-1	4.332	4.35	-18
1.553	1.555	-2	4.187	4.225	-38
1.566	1.565	1	4.106	4.225	-119
1.58	1.585	-5	4	4.037	-37

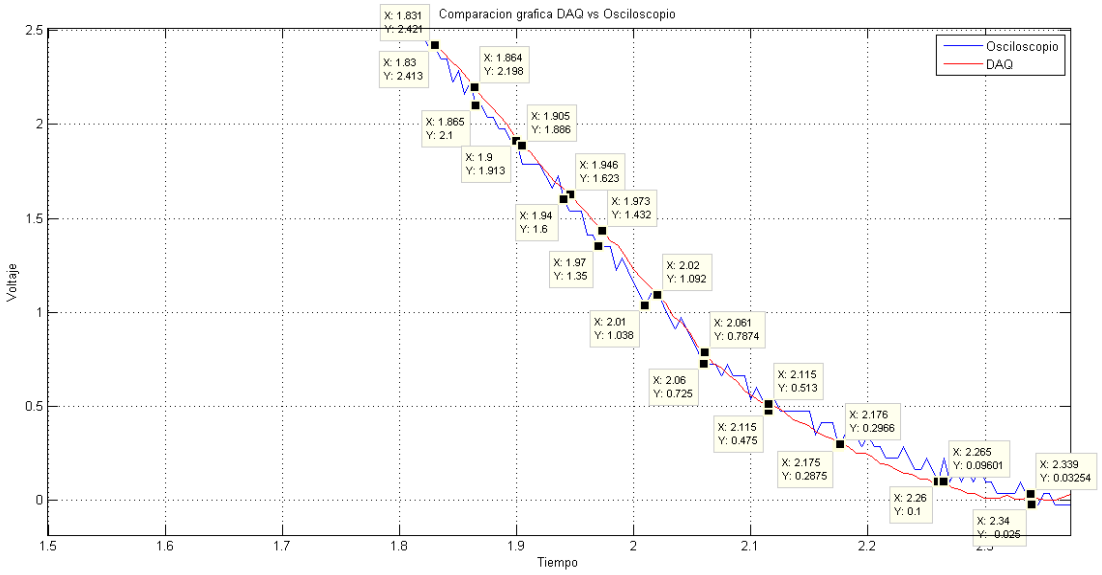


Figura 6.6.4: *Comparación de bajada.*

Tabla de datos

Tiempo DAQ	Tiempo Osciloscopio	Diferencia (ms)	Valor DAQ	Valor Osciloscopio	Error (mV)
1.831	1.83	1	2.421	2.413	8
1.864	1.865	-1	2.198	2.1	98
1.905	1.9	5	1.886	1.913	-27
1.946	1.94	6	1.623	1.6	23
1.973	1.97	3	1.432	1.35	82
2.02	2.01	10	1.092	1.038	54
2.061	2.06	1	0.7874	0.725	62.4
2.115	2.115	0	0.513	0.475	38
2.176	2.175	1	0.2966	0.2875	9.1
2.265	2.26	5	0.09601	0.1	-3.99
2.339	2.34	-1	0.03254	0.025	7

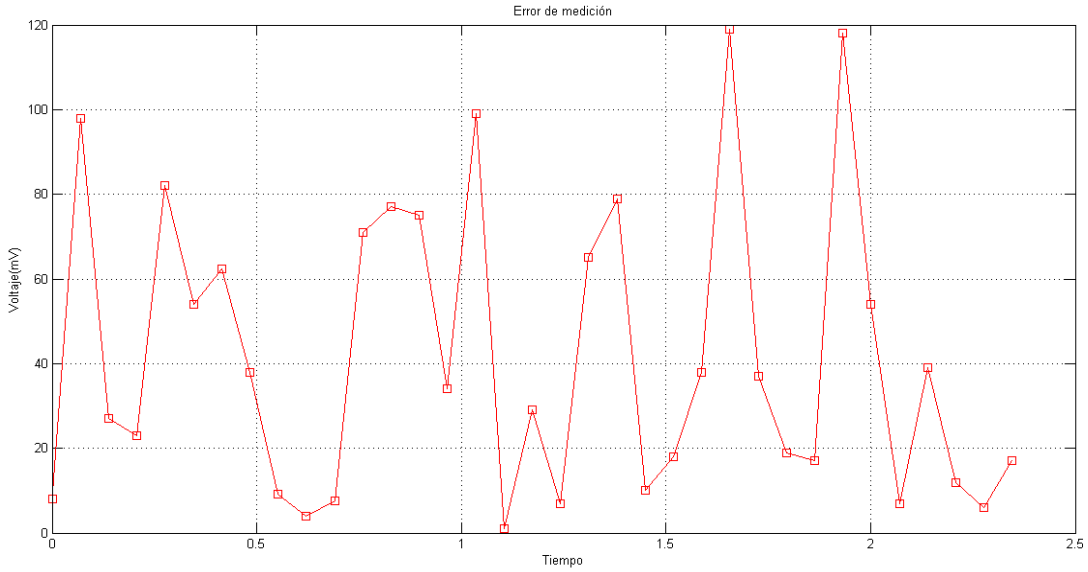


Figura 6.6.5: *Gráfica del error de medición.*

Error promedio:

$$error = \left| \frac{\sum error}{n} \right|$$

n-> Número de muestras

$$error = \frac{1461.03}{35} = 41.7437142857mV$$

El error promedio es menor de 42mV, por lo tanto se puede confiar en el DAQ para calcular el modelo de la planta.

6.7 Proceso de identificación de la función de transferencia

Para encontrar el modelo se utilizó el programa matlab versión 2013b, este permite identificar la función de transferencia a partir de una gráfica.

El primer paso fue obtener la gráfica de la respuesta al escalón de la planta utilizando el DAQ, para ello se utilizó el *instrument control toolbox* de Simulink.

La Figura 6.7.1 muestra los bloques contenidos dentro del *instrument control toolbox.*

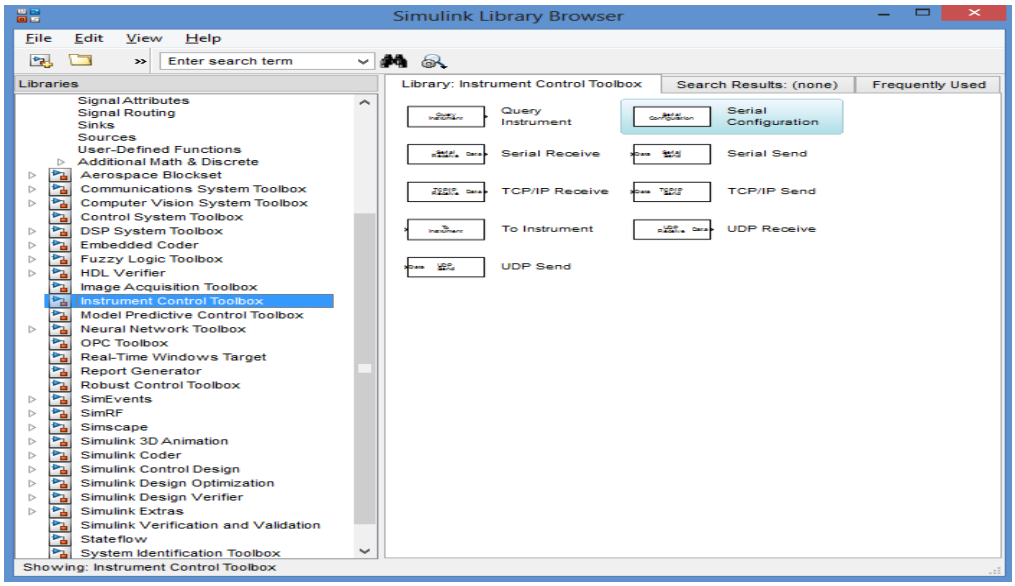


Figura 6.7.1: *Simulink instrument control toolbox.*

Para construir el modelo se utilizaron los siguientes bloques:

Serial configuration .- Define el puerto de comunicación y se definen los parámetros de comunicación con el DAQ.

- Serial receive.- Bloque encargado de recibir los datos enviados por el DAQ.
- Demux.- Demultiplexor 1 a 2.
- Scope.- Gráfica los datos enviados por el DAQ.
- Out1.- Guarda los datos recibidos por el DAQ en una matriz.

A continuación se muestra el modelo construido con los bloques mencionados anteriormente:

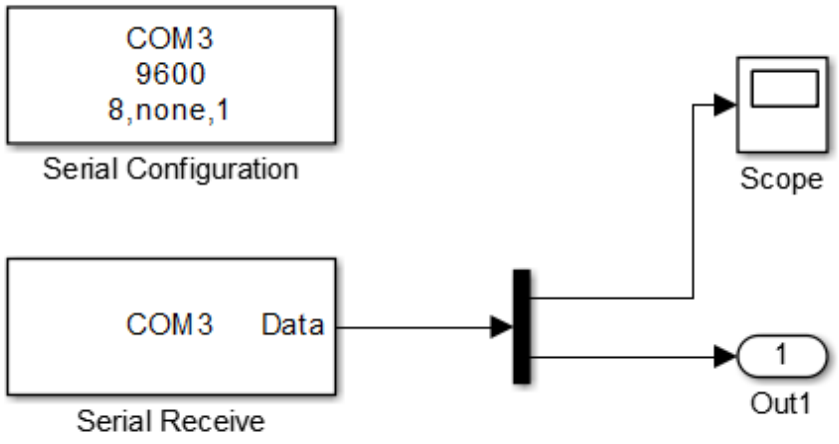


Figura 6.7.2: *Modelo del sistema de comunicación.*

Una vez construido el modelo, el siguiente paso fue conectar el DAQ al puerto de comunicación serie, el bloque encargado de recibir los datos espera el primer dato en buffer para iniciar el proceso de recepción. Los datos recibidos por el DAQ se muestran en la gráfica de la Figura 6.7.3.

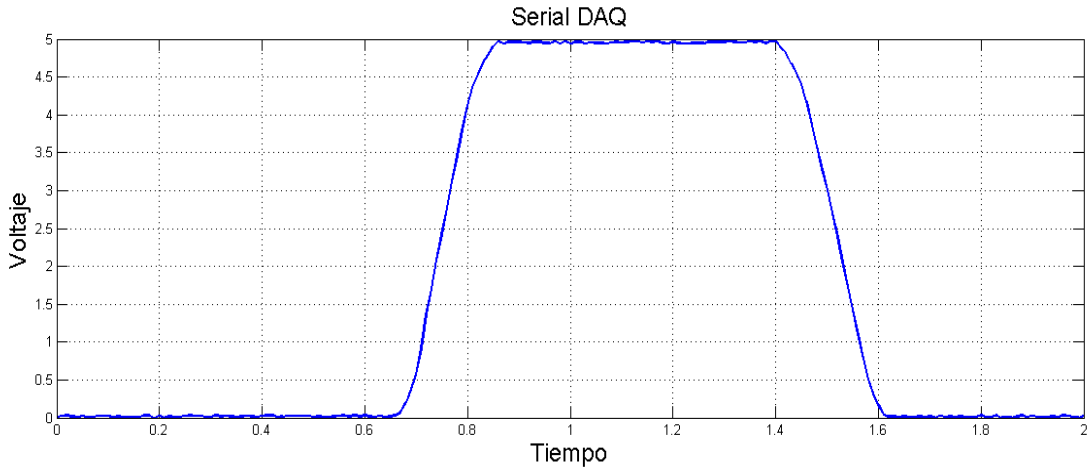


Figura 6.7.3: *Datos recibidos por el DAQ.*

6.8 Actualización del firmware del DAQ

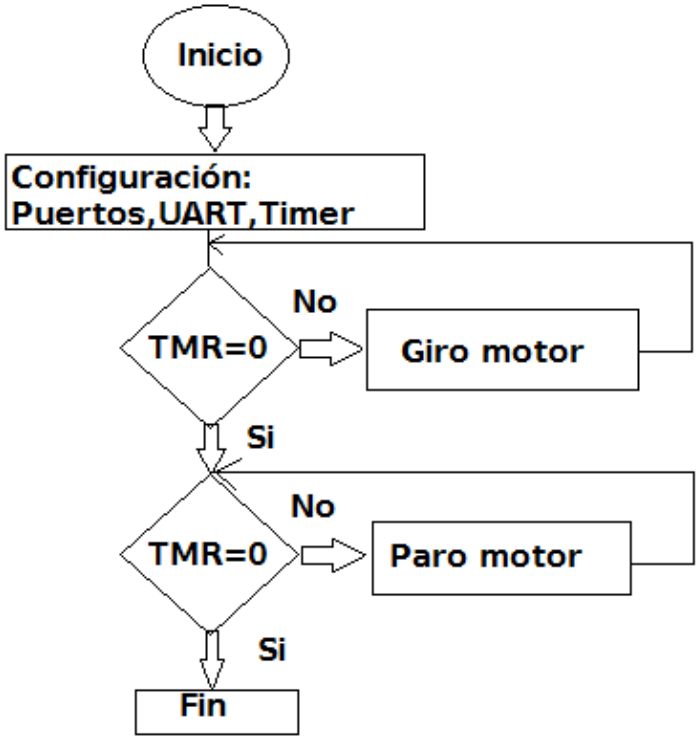
Es claro que el firmware utilizado para hacer las comparaciones no iba a ser válido para encontrar la función de transferencia de la planta, por lo tanto, era necesario hacer ajustes, el primer ajuste es que ya no es necesario regresar el motor a su posición inicial, dicho de otra forma, no es necesario llevarlo de 5Volts a 0Volts.

Observando la señal que arroja Simulink, el cambio más importante fue optimizar el código, por lo que se puede observar de la Figura 6.7.3, Simulink empieza a registrar los datos del DAQ, pero aproximadamente es hasta los 600ms que el DAQ empieza a mandar los datos, una razón pudiera ser que lo que está registrando Simulink es el ruido inducido, también se debe tomar en cuenta que como el firmware utiliza iteraciones y comparaciones para establecer el tiempo de giro del motor, esto representa tiempo, si a eso le sumamos el código que añade ruido y el tiempo que le toma sumar el ruido a la señal de salida resulta en un código no apto para ser utilizado por software especializado como lo es Simulink, por lo tanto se va a pasar de iteraciones a la utilización del *timer* del microcontrolador.

La ventaja que presenta la utilización del *timer* es muy simple, con su utilización se consiguen tiempos de ejecución controlados, dicho de otra manera, se establecen bloques de acciones para ser ejecutados en un periodo de tiempo máximo.

Para el nuevo firmware se definió un tiempo de ejecución por bloque máximo de 500ms, este tiempo puede ser establecido a cualquier otro, se tomó .5s para fines de prueba, en la Figura 6.8.1 se muestra el diagrama de flujo.

Diagrama de flujo del nuevo firmware del DAQ



6.8.1: *Diagrama de bloques.*

Una vez actualizado el firmaware del DAQ se procede a recibir los datos mediante el *toolbox de comunicación* de Simulink, la Figura 6.8.2 muestra la salida.

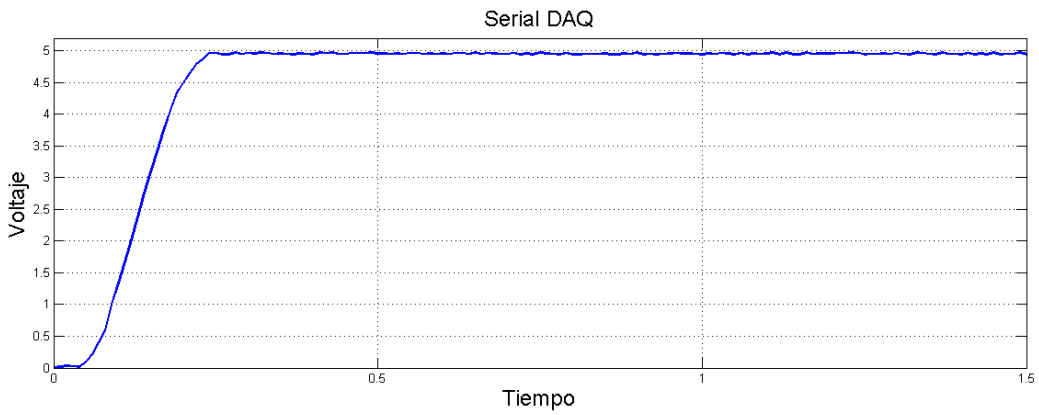


Figura 6.8.2: *Datos recibidos por el DAQ.*

En la Figura 6.8.2 se puede observar tres bloques de datos de 500ms cada uno, si la comparamos con la figura anterior (Figura 6.7.3), este nuevo firmware esta optimizado para ser utilizado con el software de Matlab.

Cabe resaltar que la utilización de un bootloader permite añadir nuevas características al dispositivo, permite concentrarse únicamente en el diseño del firmware sin la necesidad de añadir nuevos componentes al hardware, o bien sin la necesidad de crear un algoritmo para la adición de nuevas funciones en el firmware.

Una vez que se tienen los datos arrojados por el DAQ se utilizó la herramienta de identificación de sistema para encontrar la función de transferencia de la planta.

La herramienta se puede invocar mediante el comando *ident.*

fx>>ident

La herramienta de identificación requiere que se especifique el número de polos y zeros, para encontrar el número exacto se utilizó la metodología dada en Control Tutorials for Matlab & Simulink, el cual parte del modelo electromecánico del motor.

6.9 Modelo electromecánico de la planta

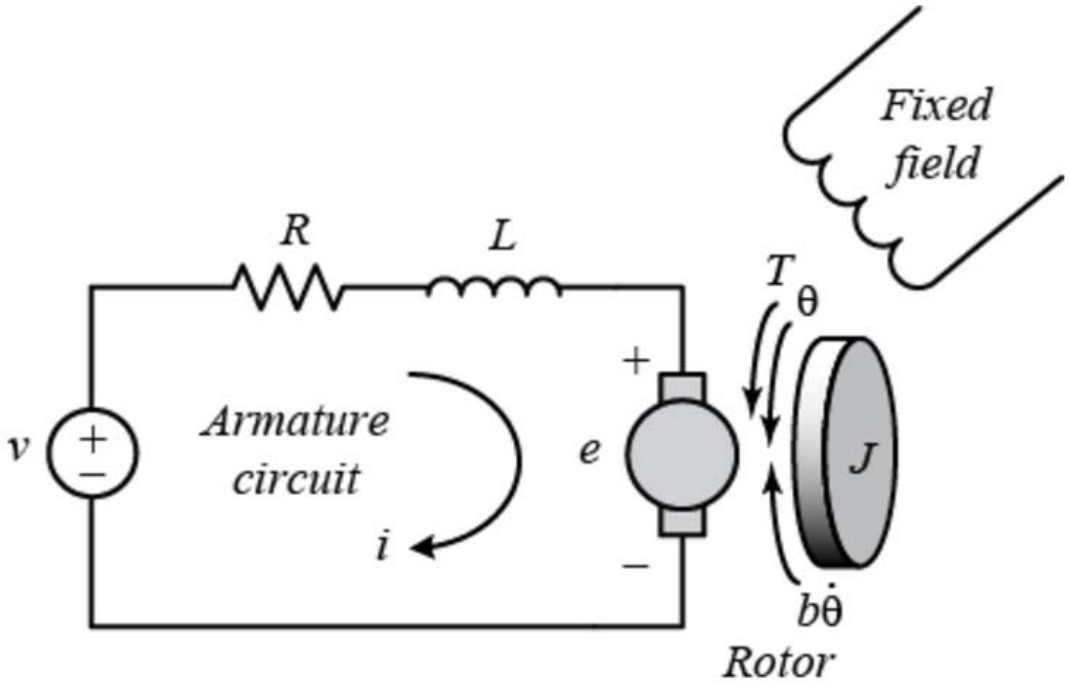


Figura 6.9.1: *Modelo electromecánico.*

Se define el torque como la corriente de la armadura multiplicado por el factor de proporcionalidad Kt:

$$T = iK_t[N.m * A] \qquad \text{… ecuación 1}$$

Se define la fuerza electromagnética o back-emf por sus siglas en inglés como una relación lineal con la velocidad angular multiplicada por un factor Kb:

$$e = \dot{\theta} K_b[\frac{rad * V}{seg}] \qquad \text{… ecuación 2}$$

Utiliza la ley de voltaje de Kirchhoff y la segunda ley de Newton para obtener la siguiente ecuación:

$$K_t = K_b$$
$$J\ddot{\theta} + b\dot{\theta} = Ki \qquad \text{… ecuación 3}$$
$$L\frac{di}{dt} + Ri = V - K\dot{\theta}$$

Se aplica la transformada de Laplace

$$s(Js+b)\theta(s) = KI(s)$$
$$(Ls+R)I(s) = V(s) - Ks\theta(s) \qquad \text{… ecuación 4}$$

Encuentra la función de transferencia de lazo abierto para velocidad

$$P(s) = \frac{\dot{\theta}(s)}{V(s)} = \frac{K}{(Js+b)(Ls+R)+K^2}\left[\frac{\frac{rad}{seg}}{V}\right] \qquad \text{… ecuación 5}$$

Por último la función de transferencia de lazo abierto para posición

$$P(s) = \frac{\theta(s)}{V(s)} = \frac{K}{s((Js+b)(Ls+R)+K^2)}\left[\frac{rad}{V}\right] \qquad \text{… ecuación 6}$$

6.10 Utilización de matlab para la identificación de la planta

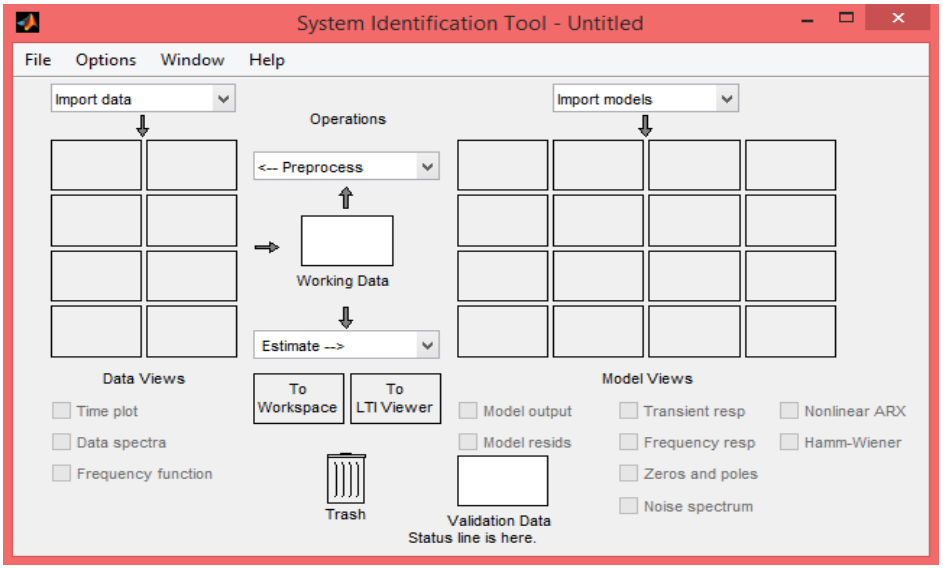


Figura 6.10.1: *Ventana principal de la herramienta de identificación.*

Como la variable de interés es la posición, puesto que el potenciómetro hace una trayectoria que va de 0V a 5V, u otra forma de verlo sería que hace un trayecto de 0 a 180 grados, de las ecuaciones del modelo electromecánico se sabe que los valores requeridos son 3 polos y ningún zero.

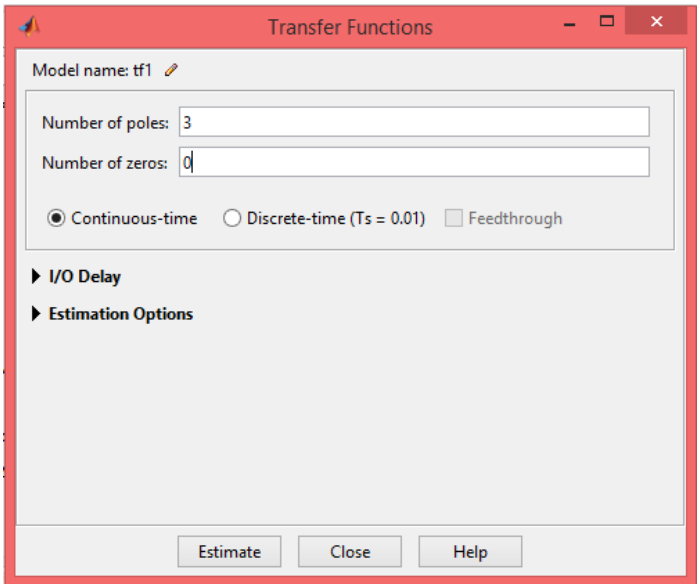


Figura 6.10.2: *Definiendo polos y zeros.*

6.10.1 Resultados de la identificación

Una vez establecidos (s) el número de polos y zeros, el programa empieza a estimar la función de transferencia de la planta, una vez que el programa finaliza los cálculos muestra el porcentaje de concordancia, para este caso fue del 98.22%, cabe señalar en este punto que gracias a la adición de ruido y a la adecuación del firmware del DAQ, se obtiene este porcentaje de concordancia, la Figura 6.10.3 muestra el resultado de la identificación:

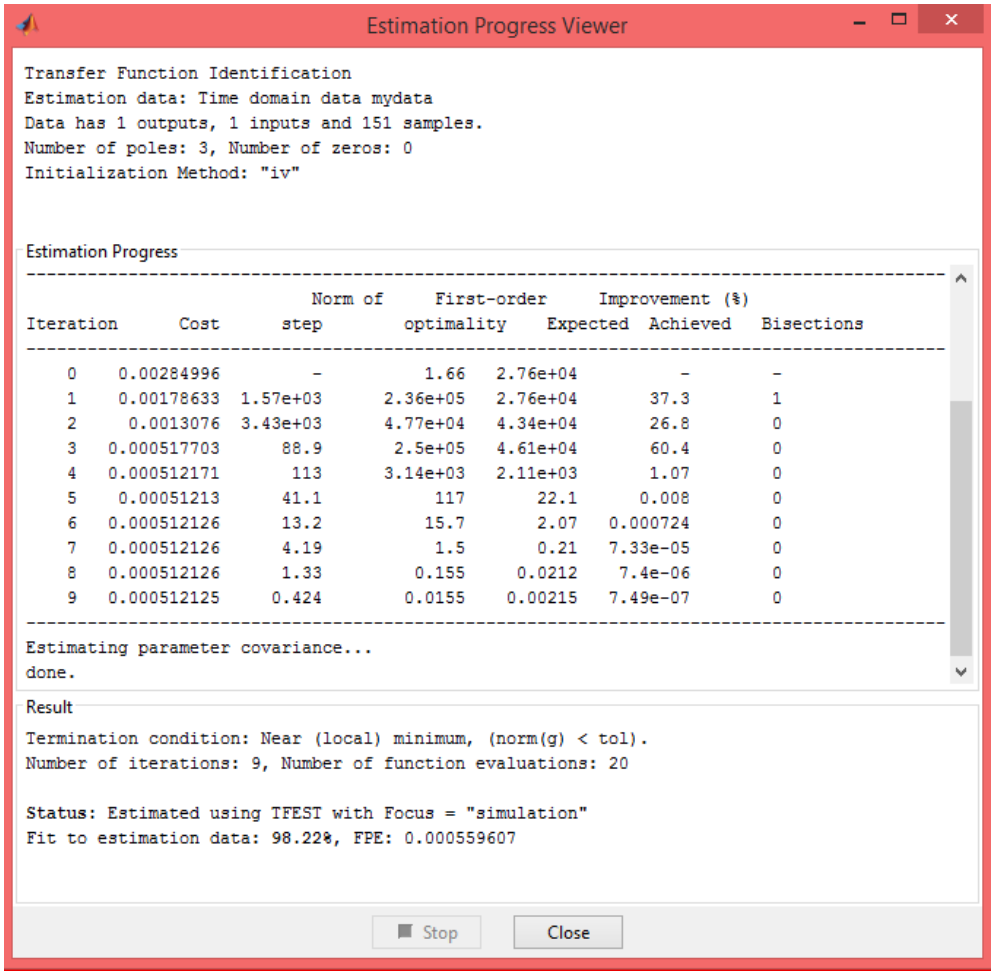


Figura 6.10.1.1: *El resultado tiene un 98.22% de concordancia con la señal física.*

Para comprobar el modelo obtenido se va a comparar con la señal física contra el modelo calculado por el software de identificación, la Figura 6.10.1.2 muestra ambas señales:

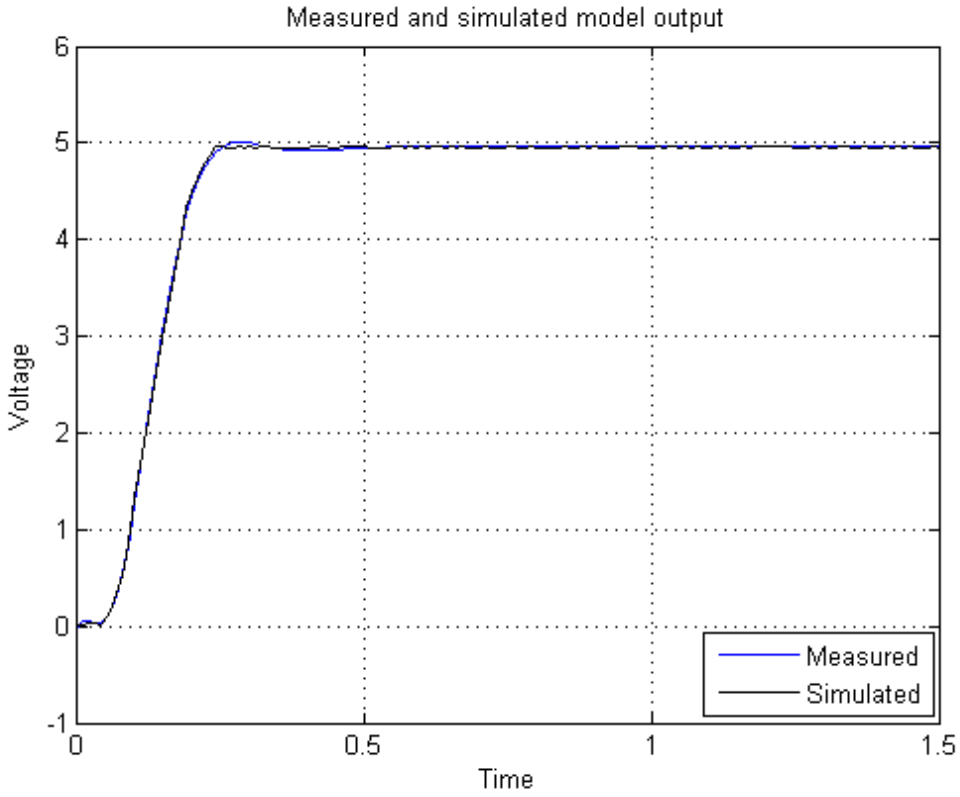


Figura 6.10.1.2: *Comparación de la señal física con el modelo encontrado.*

La función de transferencia es la siguiente:

$$tf = \frac{3134}{s^3 + 41.79s^2 + 962.7s + 8722}$$

6.11 Transformada Z del controlador PID

Una vez que se obtuve el modelo de la planta, el siguiente paso fue diseñar el controlador digital, partiendo de la estructura paralela del control PID y a partir de ella se calculó la transformada Z.

Forma paralela

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

Aplicando la transformada de Laplace

$$\frac{H(s)}{e(s)} = K_p + \frac{K_i}{s} + K_d s$$

Aplicando la transformada Z

$$\left.\frac{H(s)}{e(s)}\right|_{s=\frac{1-z^{-1}}{T}} = K_p + \frac{K_i T}{1-z^{-1}} + \frac{K_d (1-z^{-1})}{T}$$

Para facilitar los cálculos se definió las ganancias de la siguiente manera

$$K_p = K_p$$
$$K_i = K_i T$$
$$K_d = \frac{K_d}{T}$$

Desarrollando con este nuevo formato se obtiene la siguiente ecuación:

$$\frac{H(z)}{e(z)} = \frac{K_p + K_i T + \frac{K_d}{T} + z^{-1}(-K_p - 2(\frac{K_d}{T})) + \frac{K_d}{T} z^{-2}}{1-z^{-1}}$$

Para poder implementar esta ecuación en un microcontrolador es necesario pasarla a su forma diferencial

Se definen:

$$K1 = K_p + K_i T + \frac{K_d}{T}$$
$$K2 = -K_p - 2(\frac{K_d}{T})$$
$$K3 = \frac{K_d}{T}$$

Por lo tanto la ecuación (1) se puede escribir de la siguiente manera:

$$\frac{H(z)}{e(z)} = \frac{K1 + K2z^{-1} + K3z^{-2}}{1 - z^{-1}}$$

Desarrollando la ecuación

$$H(z)(1 - z^{-1}) = (K1 + K2z^{-1} + K3z^{-2})e(z)$$
$$H(z) - H(z)z^{-1} = (K1 + K2z^{-1} + K3z^{-2})e(z)$$
$$H(z) - H(z)z^{-1} = e(z)K1 + e(z)K2z^{-1} + e(z)K3z^{-2}$$

z^-n representa un retardo en tiempo, por lo tanto para pasar de la transformada. Z al tiempo utilizo una ***k*** para representarlo:

$$z^0 = k$$
$$z^{-1} = k - 1$$
$$z^{-2} = k - 2$$

Sustituyendo:

$$H(k) - H(k-1) = e(k)K1 + e(k-1)K2 + e(k-2)K3$$
$$H(k) = H(k-1) + e(k)K1 + e(k-1)K2 + e(k-2)K3$$

Por último se restablecieron los valores de K1,K2 y K3 para obtener la ecuación final.

$$H(k) = H(k-1) + e(k)(K_p + k_i T + \frac{K_d}{T}) + e(k-1)(-k_p - 2(\frac{K_d}{T})) + e(k-2)\frac{K_d}{T}$$

6.12 Modelo del controlador en simulink

Para modelar y encontrar las ganancias para los parámetros k1, k2 y k3 se utilizó el programa simulink versión 8.2.

Se establecieron los parámetros necesarios del modelo en un archivo m.

```
%E. David Rojas Serrano <edx@edx-twin3.org>

%Variables para el modelo
clc
%K1 = Kp+(Ki*Ts)+(Kd/Ts);
Kp=0.363;
%K2 = -Kp-2*(Kd/Ts);
Ki=1.277;
%K3 = Kd/Ts;
Kd=0;
%Tiempo de muestreo
Ts = 0.01;
```

Cabe señalar que los valores para los parámetros Kd, Ki y Kp se eligieron de forma totalmente aleatoria, prácticamente es para que el programa de Simulink pudiera simular la respuesta del controlador con estos valores para posteriormente ajustarlos, mientras que el parámetro Ts corresponde al tiempo de muestreo, el cual tiene un valor de 10 ms, lo que corresponde a 100Hz, dicho de otra manera toma 100 muestras cada segundos, este valor fue elegido basado en el tiempo de ejecución del microcontrolador, por ejemplo, para una frecuencia de 1KHz se tiene un periodo de 1ms, si se toma en cuenta el tiempo que le toma al ADC cuantificar la señal de entrada, más el tiempo en los cálculos de la señal de control,

más el tiempo para adecuar la señal de control, el microcontrolador pudiera no ser capaz de cumplir con este tiempo de muestreo.

Tomemos como ejemplo el siguiente código, el cual consiste en multiplicar dos números fraccionarios, y redondear el resultado de la operación:

```
while(1)
{
   a=1.2;
   b=0.4;
   c=a*b;
   d=(int) c;
   for(;;){}
}
```

El tiempo de ejecución de este código para el PIC16F1619 @ 8Mhz es de:

Stopwatch = **1.109 ms**

Más de un milisegundo, que era el tiempo de muestreo para 1KHz, cabe destacar que estas operaciones son utilizadas a la hora de calcular la señal de control, una solución sería incrementar la frecuencia del reloj, con lo cual pudiera ser o no necesario un cristal externo, lo que significa, calcular los capacitores adecuados para trabajar el microcontrolador a frecuencias elevadas (por encima de los 20MHz), por lo tanto, a menos que sea absolutamente necesario tiempos de muestreo tan rápidos y como esta tesis no tiene como fin una aplicación en específico, se eligió un tiempo de muestreo de 10ms.

A partir de la ecuación que se calculó, se diseñó la estructura del controlador PID:

$$H(k) = H(k-1) + e(k)(K_p + k_i T + \frac{K_d}{T}) + e(k-1)(-k_p - 2(\frac{K_d}{T})) + e(k-2)\frac{K_d}{T}$$

La planta utiliza la función de transferencia que se encontró en el apartado *modelación matemática de la planta*:

$$tf = \frac{3146}{s^3 + 41.87s^2 + 965.4s + 8757}$$

Modelo final del sistema:

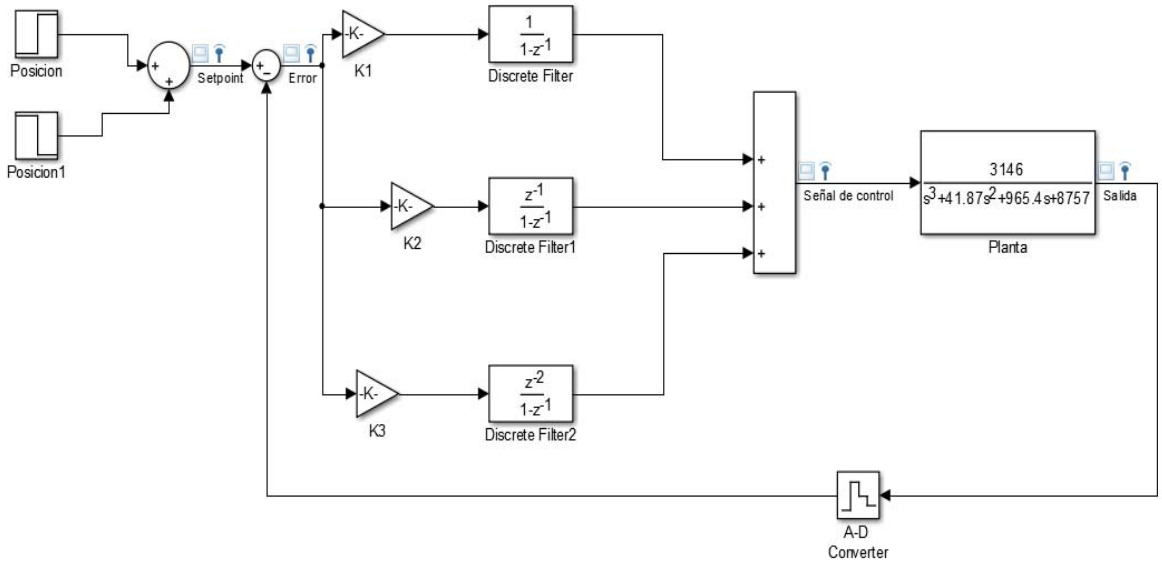


***Figura 6.12.1**: Modelo para Simulink del sistema.*

6.13 Respuesta del sistema no sintonizado

El setpoint de posición que va a entrar al sistema para ver su comportamiento se definió para que fuera de 0 a 180 grados (0V-5V) y de 180 a 0 grados (5V-0V).

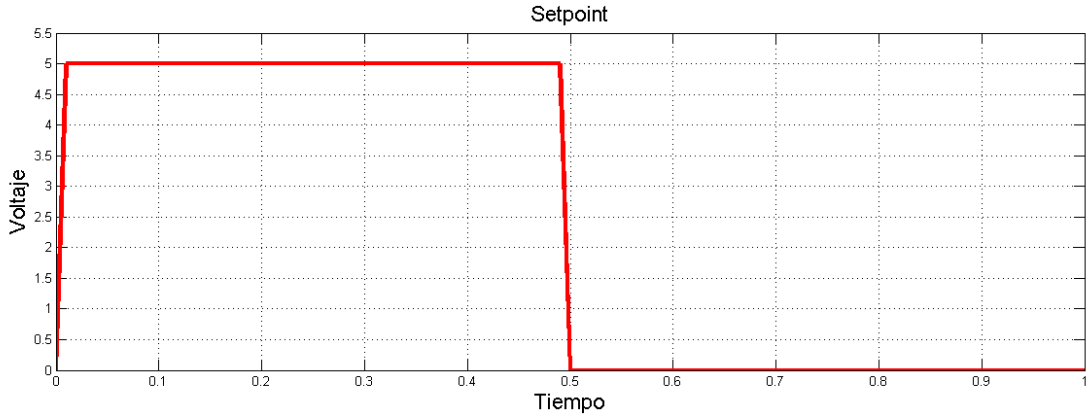


***Figura 6.13.1**: Setpoint de entrada al sistema.*

La Figura 6.13.2 muestra el setpoint (rojo) vs la respuesta del controlador (azul)

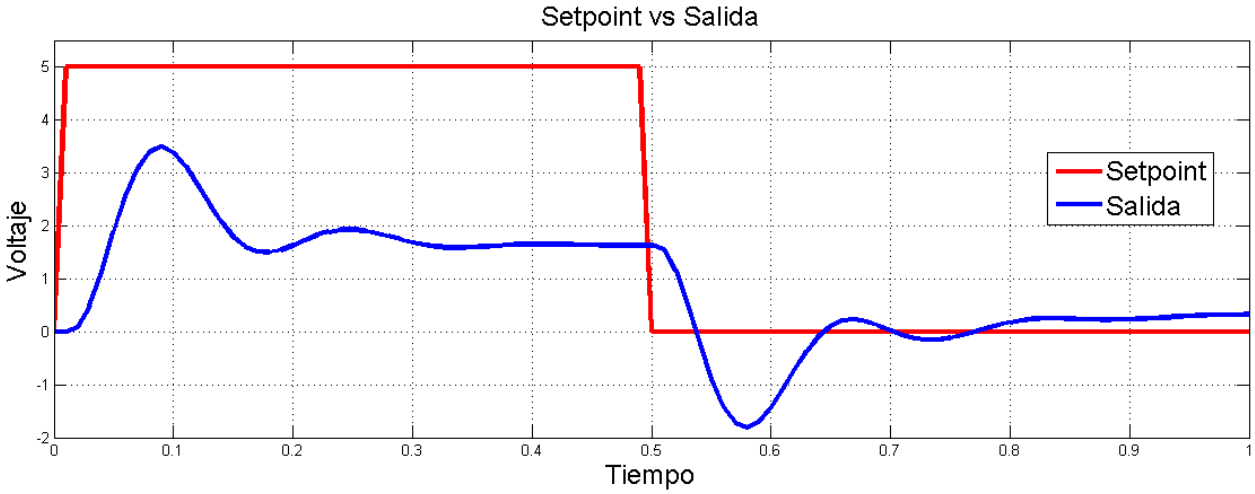


***Figura 6.13.2**: Setpoint vs respuesta del controlador.*

La Figura 6.13.3 muestra la señal de control

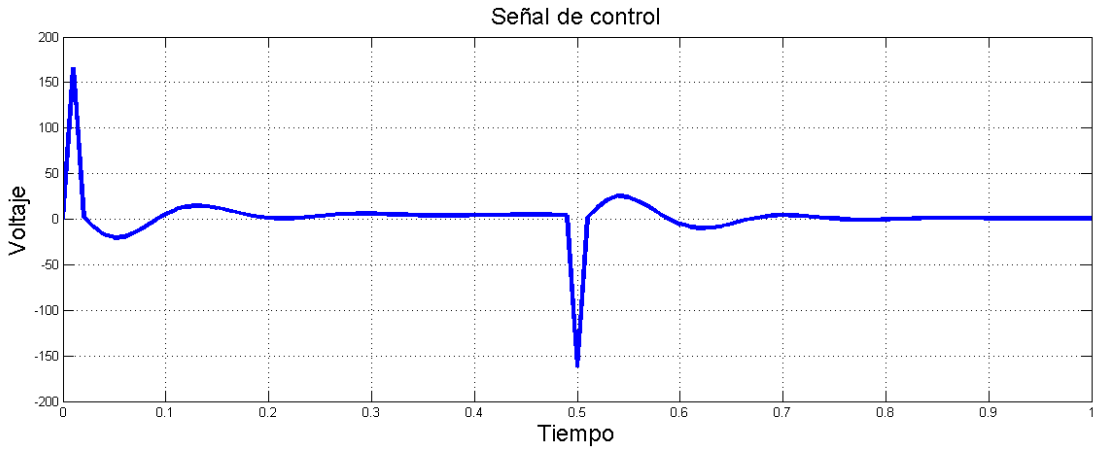


***Figura 6.13.3**: Señal de control.*

La Figura 6.13.4 muestra la señal del error.

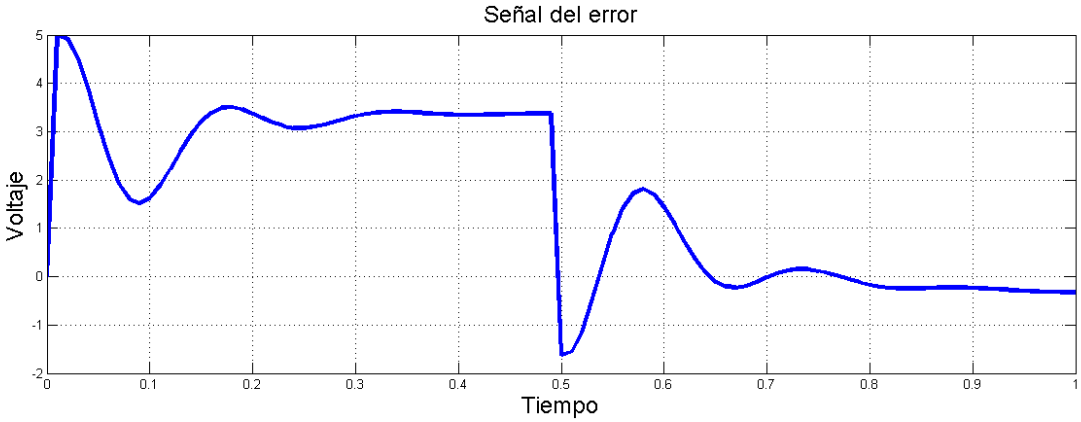
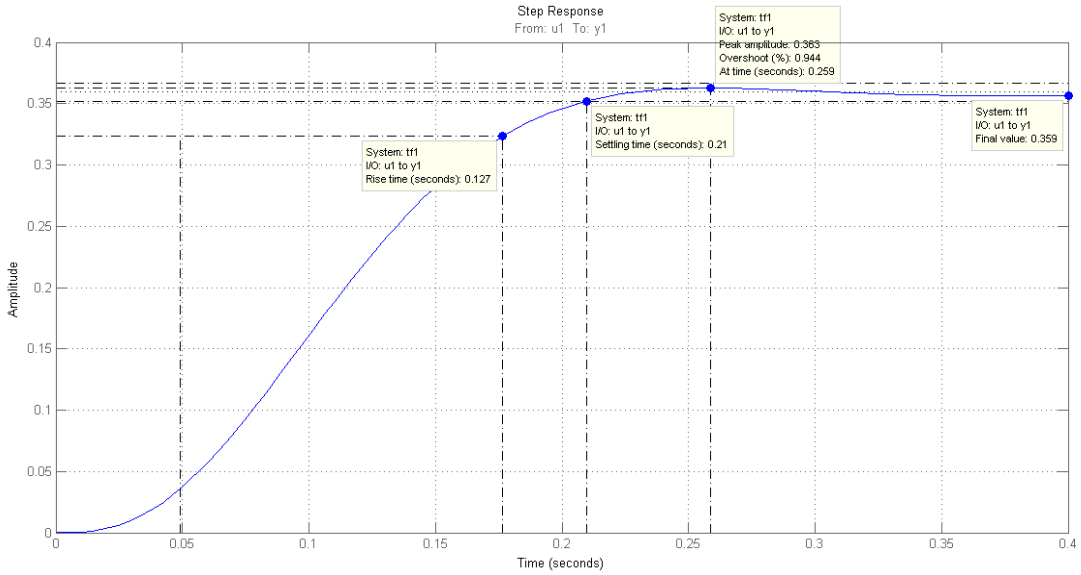


***Figura 6.13.4**: Señal de error.*

6.14 Sintonización del sistema

Para poder sintonizar el controlador es necesario conocer los parámetros temporales de desempeño del motor, estos parámetros corresponden al tiempo de subida, máximo sobrepaso y tiempo de asentamiento.

5.14.1 Parámetros de desempeño



Gráfica 6.14.1.1: *Respuesta al escalón de la planta (Motor de cd con escobillas).*

Tiempo de subida	0.127 segundos
Tiempo de asentamiento	0.21 segundos
Sobrepaso	0.944%

Para sintonizar el controlador se utilizó la herramienta *design optimization* de simulink con los siguiente parámetros de desempeño, dado que solo se pretende demostrar el funcionamiento de la herramienta *desing optimization* para encontrar las ganancias del controlador, los parámetros de desempeños no cumplen con ningún requerimiento en específico, estos parámetros fueron propuestos para servir de ejemplo.

Definiendo parámetros de desempeño deseados: (*(Motor de cd con escobillas)*

Tiempo de subida	0.1 segundos
Tiempo de sentamiento	0.2 segundos
Máximo sobrepaso	9%

Estos parámetros son introducidos, en la ventana de la herramienta de optimización del sistema, la figura 6.14.1.2 muestra la ventana .

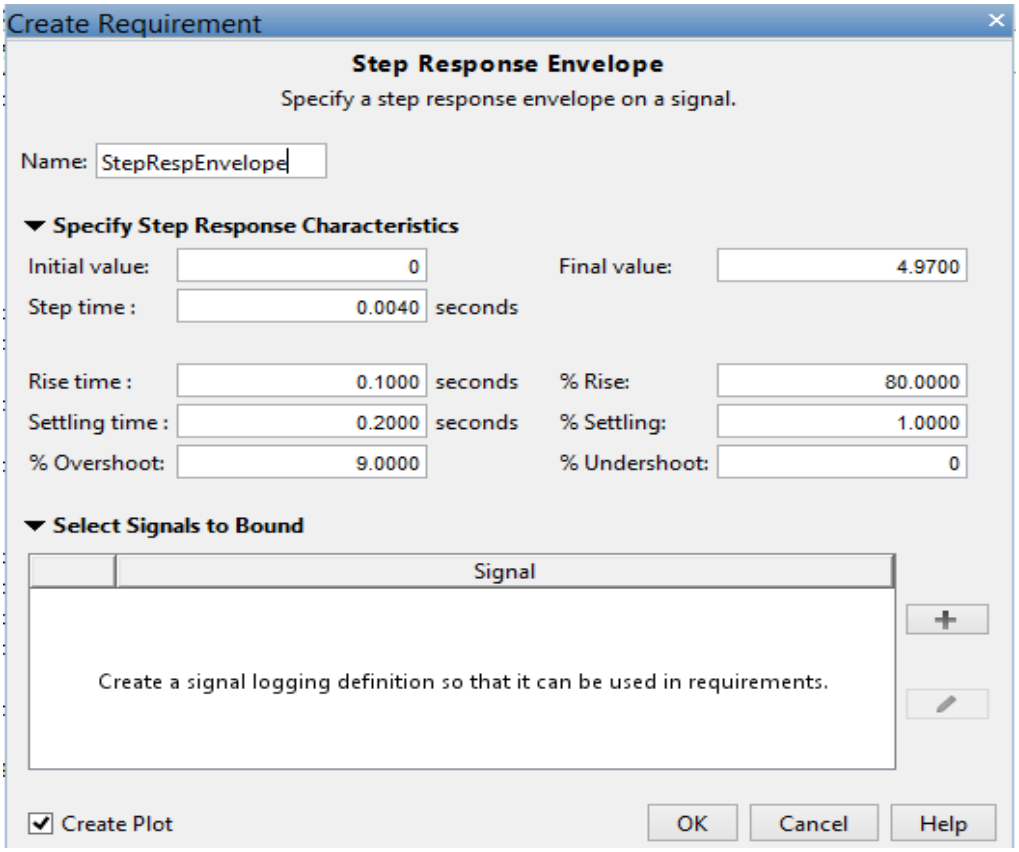


Figura 6.14.1.2: *Parámetros de desempeño.*

Una vez definidos los parámetros, se gráfica la salida actual de sistema, las líneas negras que presenta la Figura 6.14.1.3 son los parámetros de desempeño que se establecieron anteriormente, mientras que la señal azul corresponde a la salida actual del sistema.

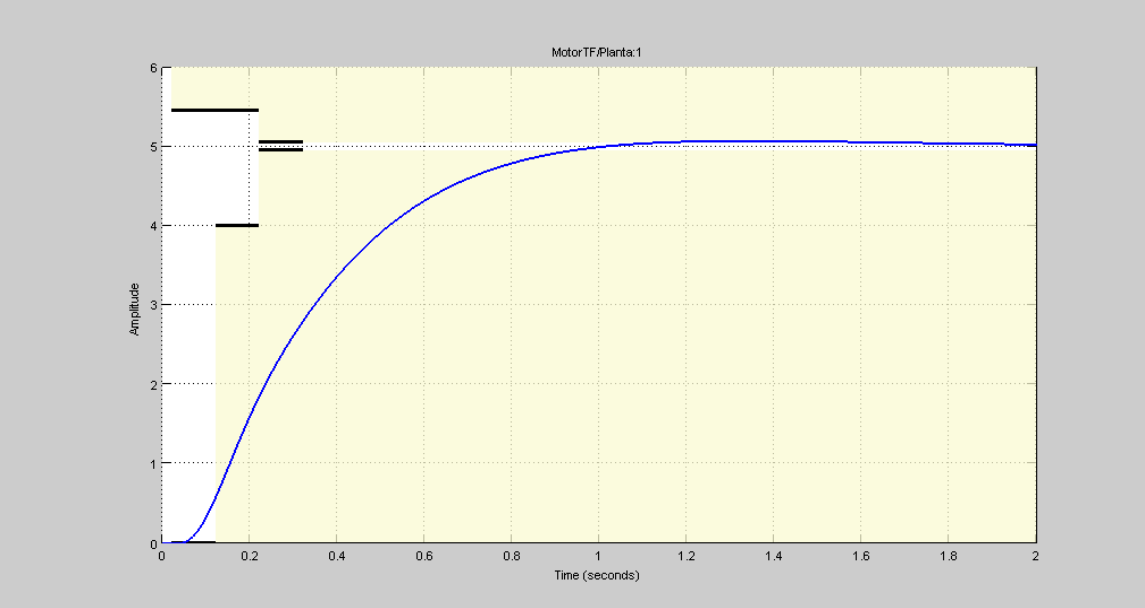


Figura 6.14.1.3 *Respuesta del sistema no sintonizado:*

La Figura 6.14.1.4 muestra el comportamiento del sistema una vez que han sido sintonizados el valor de los parámetros Kp, Ki, Kd que cumplen con los parámetros de desempeño establecidos desde un principio.

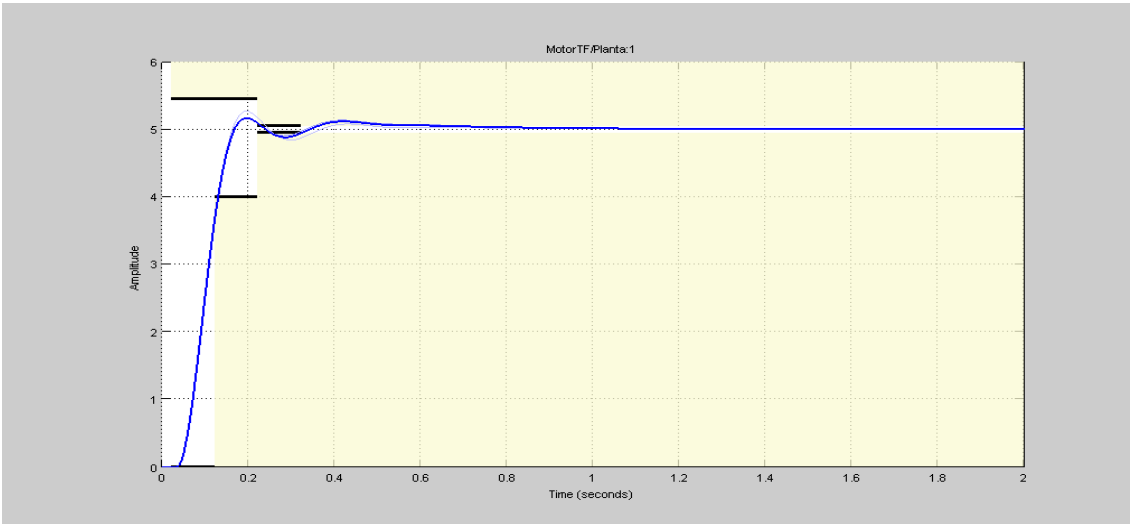


Figura 6.14.1.4: *Respuesta del sistema sintonizado.*

En la Figura 6.14.1.5 se puede observar la evolución de los valores para las ganancias de los parámetros del controlador PID.

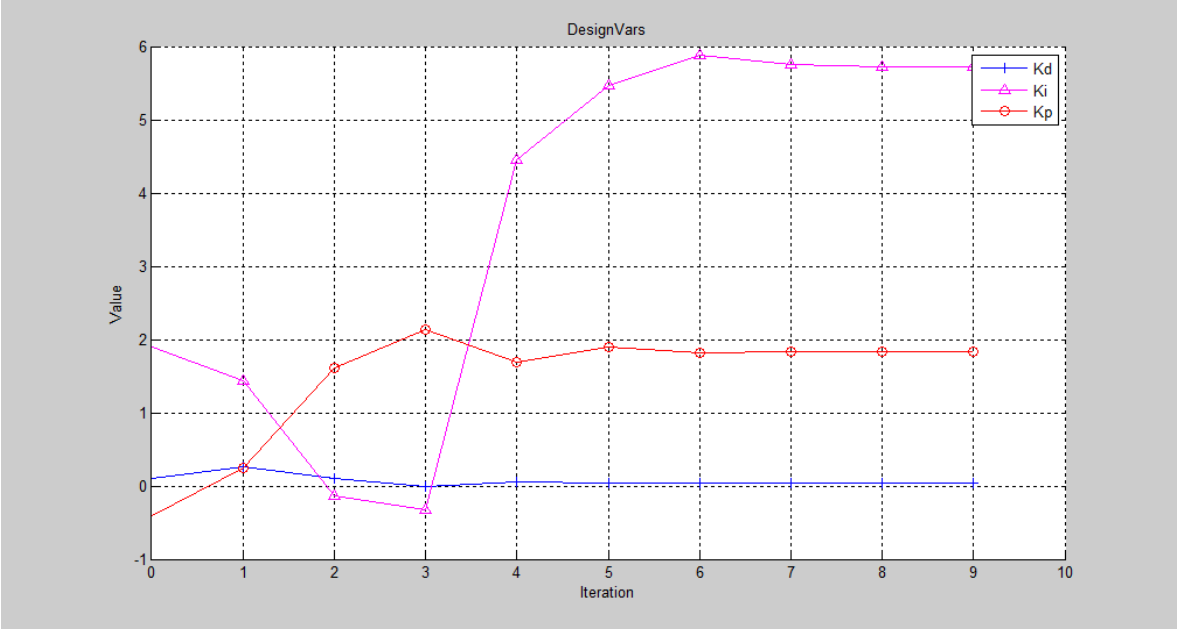


Figura 6.14.1.5: *Gráfica con los ajustes de las ganancias para los parámetros Kp, Ki, Kd.*

Los valores calculados por el programa para las ganancias son:

$$K_d = 0.0363$$
$$K_i = 5.1190$$
$$K_p = 1.6751$$

Comparación entre valores:

Valor Anterior	**Valor Actual**
Kp= 0.363	Kp=1.6751
Ki=1.277	Ki=5.1190
Kp=0	Kp=0.0363

El programa automáticamente actualiza los valores de las ganancias, por lo tanto no es

necesario establecerlos nuevamente o sobre escribirlos.

Cabe mencionar que los resultados encontrados por el programa dependerán enteramente de los parámetros de desempeño que se le especifiquen.

Una vez obtenido el valor de las ganancias el siguiente paso es ver cómo se comporta el sistema frente al setpoint.

6.15 Resultados obtenidos

La Figura 6.15.1 muestra el setpoint (rojo) vs la respuesta del controlador (azul).

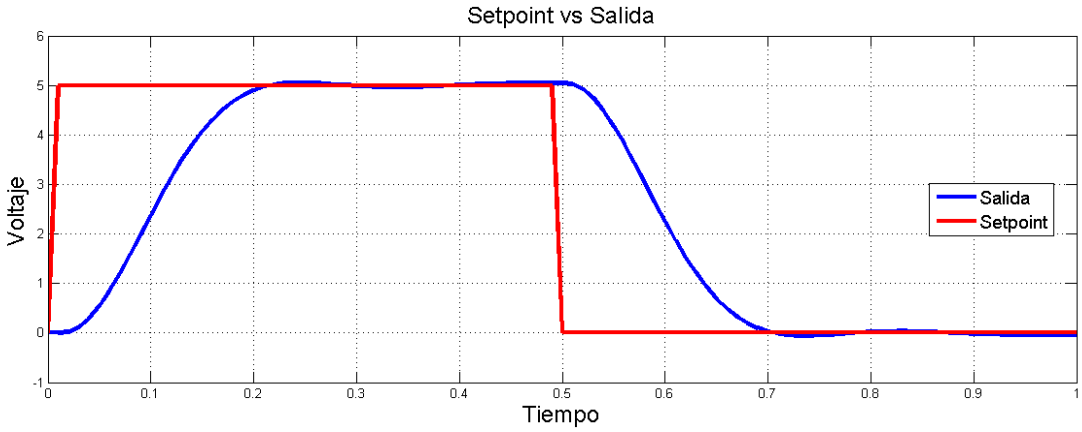


Figura 6.15.1: *Setpoint vs respuesta del controlador.*

La Figura 6.15.2 muestra la señal de control.

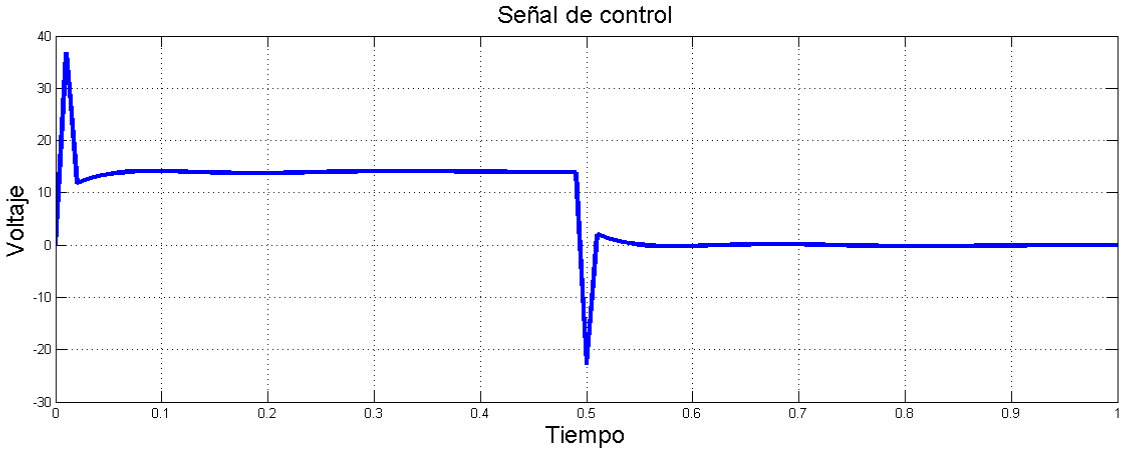


Figura 6.15.2: *Señal de control.*

La Figura 6.15.3 muestra la señal de error.

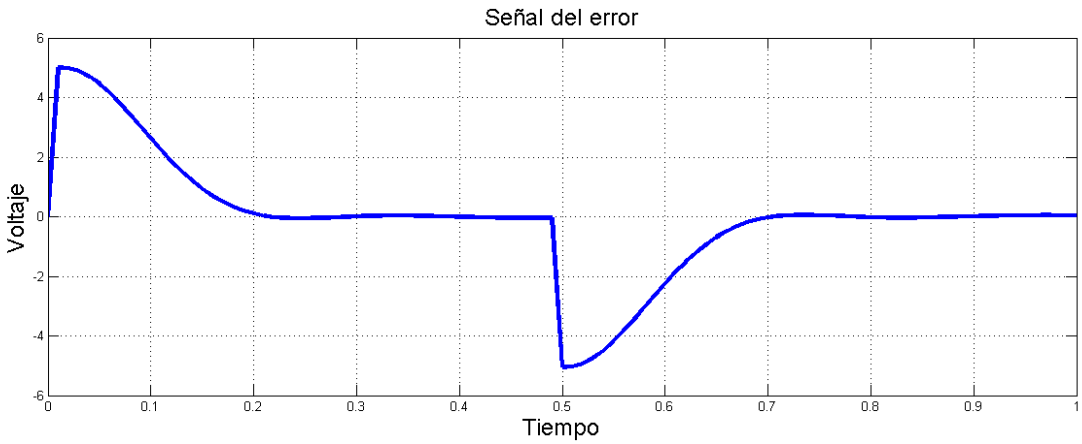


Figura 6.15.3: *Señal del error.*

Capítulo 7

Aplicación 2: Control de posición

7.1 Planteamiento del problema:

Se desea diseñar un control de posición utilizando un controlador PID, la posición que debe cubrir es de 0 a 180 grados, los parámetros de desempeño se dejan a criterio del estudiante.

7.2 Desarrollo

Aunque la ecuación del PID puede ser implementada por software, se utilizó las características del microcontrolador PIC16F1619, la familia de microcontroladores PIC16F161X de microchip está diseñada para proporcionar capacidades avanzadas de control de motores, entre las cuales destaca el módulo especializado para hacer control digital, generador de onda complementaria, cuatro celdas lógicas programables compatibles con lógica secuencial y combinacional y control dead-band.

En las hojas de datos del microcontrolador especifican el modelo del controlador implementado en hardware el cual es la siguiente:

$$C_0[k] = C_0[k-1] + {}^0k_1e[k] + {}^0k_2e[k-1] + {}^0k_3e[k-2]$$

Dónde:

$$K1 = K_p + K_iT + \frac{K_d}{T}$$

$$K2 = -K_p - \frac{2K_i}{T}$$

$$K3 = \frac{K_d}{T}$$

Analizando la ecuación resulta ser la transformada Z del control clásico de 3 términos que se calculó en la sección *Modelo matemático*, por lo tanto no es necesario hacer cambios posteriores en los cálculos obtenidos.

Se utilizó el bootloader J16F1619 para desarrollar el firmware, la ventaja que presenta la

utilización de un bootloader en este tipo de sistemas puede ser la mejor opción y esto es debido a que es de vital importancia tomar en cuenta lo siguiente:

- El valor de las ganancias (Kd, Ki y Kp) son números signados no enteros, dicho de otra forma, sería necesario definir la longitud de bits para el número entero y la mantisa.
- Puede ser necesario hacer ajustes de las ganancias o las ecuaciones.
- Tiempo de diseño

Hasta este punto se tiene lo siguiente:

1. Ecuaciones del controlador PID discreto.
2. Modelo de la planta (función de transferencia).
3. Modelo del controlador en Simulink.
4. Optimización de los parámetros Kp, Ki, y Kd.

Por lo tanto ya solo es necesario calcular los valores de las ganancias, y adecuar la señal de control la cual se va a mandar al puente H, para controlar la posición del motor.

Antes de continuar, basado en la poca experiencia que he adquirido en las prácticas de la materia de diseño de controladores sé, que la mejor opción para el control de un motor es utilizar un controlador PI en vez de utilizar un PID, recordemos que la aparte derivativa mide que tan rápido cambia el error, entre más rápida sea la tasa de cambio, la acción derivativa va actuar contra la salida para restringirla, si el error no cambia esto puede llevar a un sobre paso o hacer inestable el sistema, por lo tanto se va a utilizar un PI.

El Cuadro 7.2.1 muestra los valores iniciales que van a servir para obtener la respuesta del sistema no sintonizado.

Valores de los parámetros Kd ,Ki y Kp	
Kp	0
Ki	0.9
Kd	1
Ts	0.01

Cuadro 7.2.1: *Valores iniciales de los parámetros del controlador.*

Primero se define el setpoint, este va de 0 grados a 180 grados, después pasa de 180 grados a 90 grados (5Volts-2.5Volts), por último pasa de 90 grados a 0 grados (0Volt-5Volts) La Figura 7.2.1 muestra el setpoint que entra al sistema:

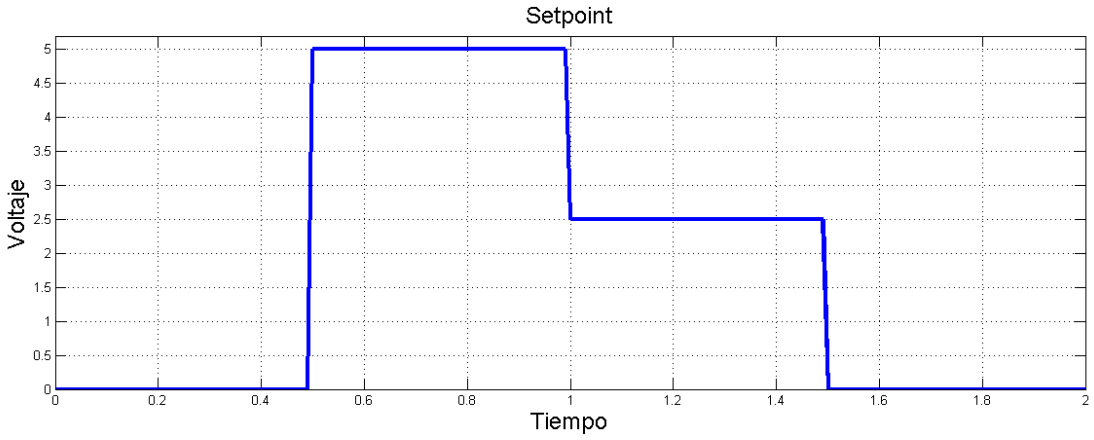


Figura 7.2.1: *Setpoint.*

La Figura 7.2.2 muestra la salida del controlador frente el setpoin.

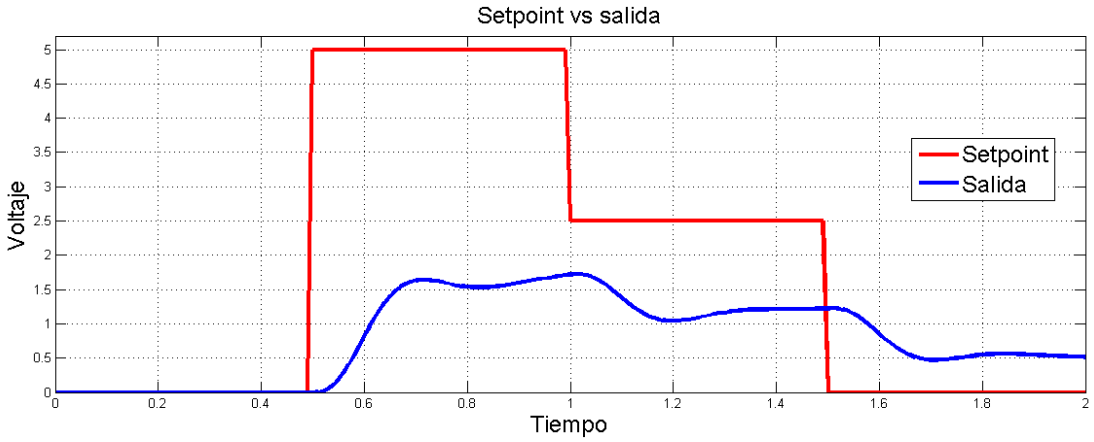


Figura 7.2.2: *Setpoint vs salida del sistema.*

La Figura 7.2.3 muestra la señal de control

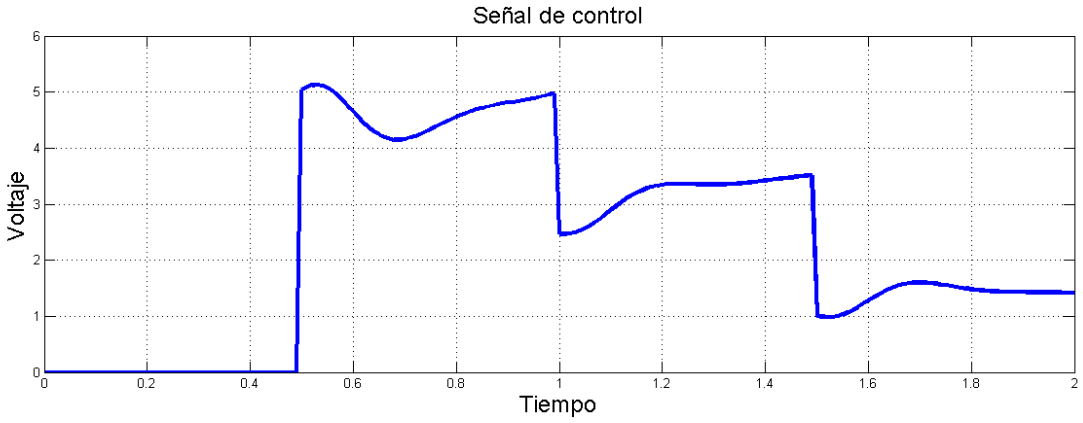


Figura 7.2.3: *Señal de control.*

La Figura 7.2.4 muestra la señal del error

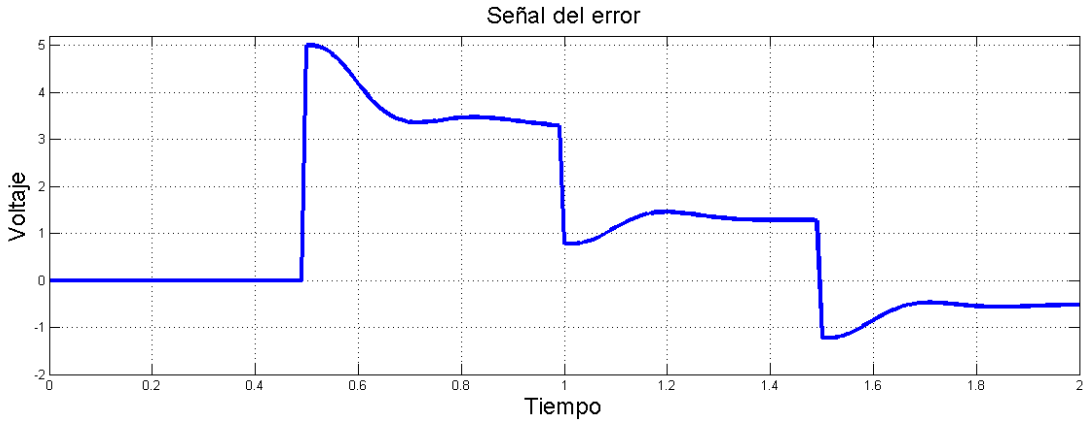


Figura 7.2.4: *Señal del error.*

Se definen los parámetros de desempeño deseados. El Cuadro 7.2.2 muestra los valores utilizados para este proyecto, cabe mencionar que estos valores no cumplen criterios o valores de desempeños en específico, se establecieron de esta forma para servir puramente de ejemplo.

Rise time [seg]	0.2207	%Rise	80
Settling time [seg]	0.3376	%Settling time	.7
% Overshoot	3.7026	%Undershoot	0

Cuadro 7.2.2: *Parámetros de desempeño.*

En la Figura número 7.2.5 se puede ver la salida actual del sistema, las líneas negras corresponden a los parámetros de desempeño que se establecieron anteriormente, mientras que la señal azul corresponde a la salida actual del sistema de control con parámetros de control iniciales.

.

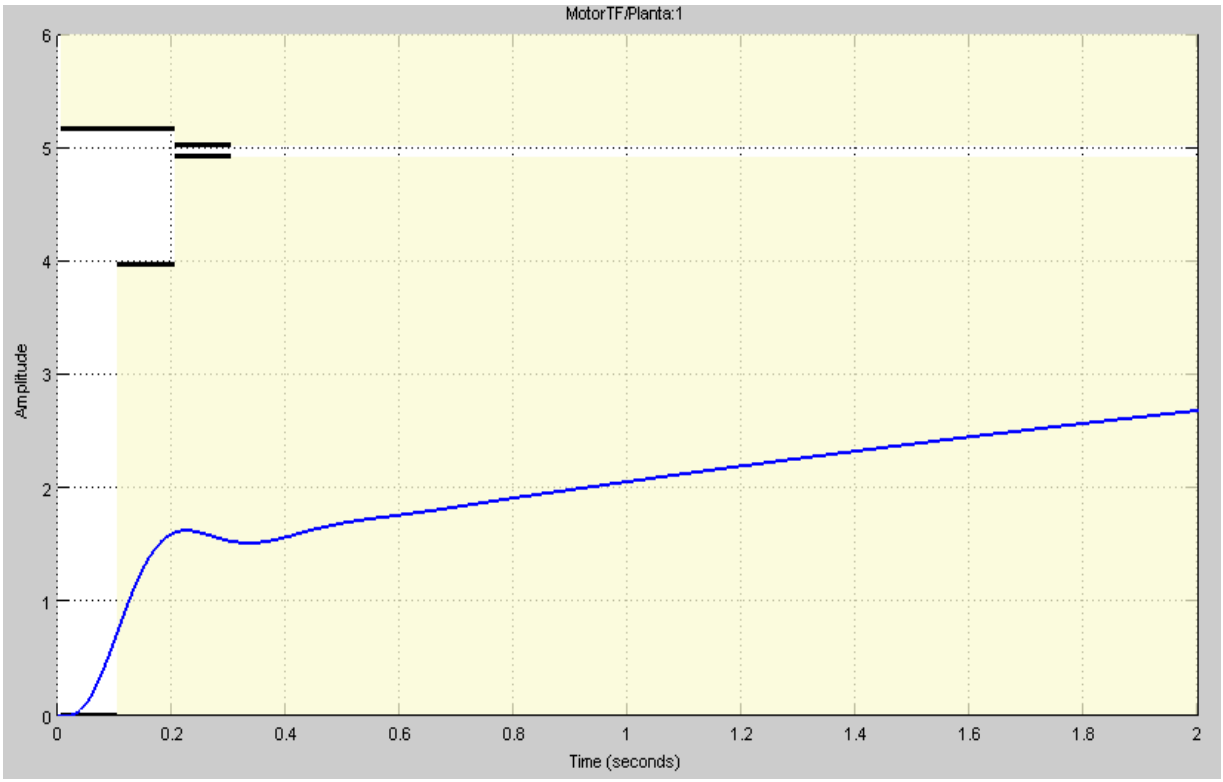


Figura 7.2.5: *Salida del sistema actual.*

En la Figura número 7.2.6 se puede observar la salida del sistema una vez que han sido sintonizados los parámetros del controlador, se puede notar como la respuesta (señal color azul) encaja en los parámetros de desempeño (líneas negras).

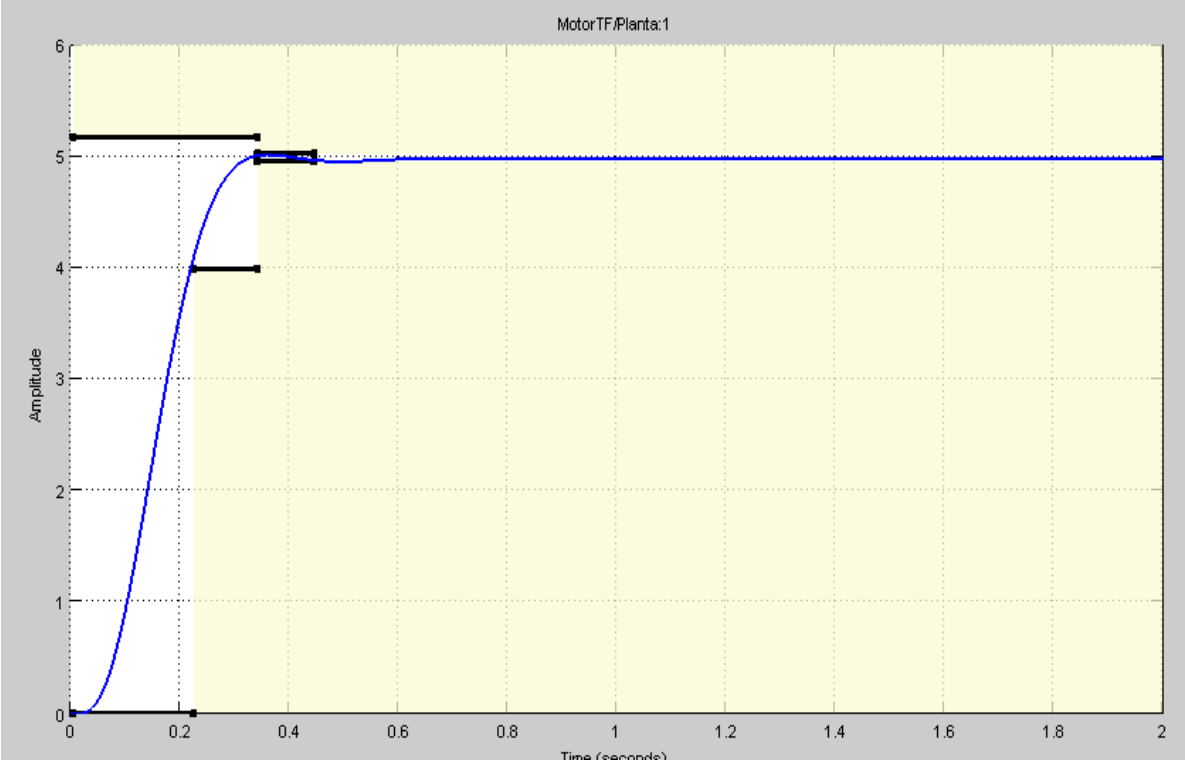


Figura 7.2.6: *Salida del sistema sintonizado.*

El Cuadro 7.2.3 muestra los valores encontrados para los parámetros Kp, Ki y Kd

Valores de los parámetros Kd, Ki y Kp	
Kd	0
Ki	17.811992
Kp	0.79190

Cuadro 7.2.3: *Valores de los parámetros Kd, Ki y Kp .*

Para comprobar los valores encontrados por el software, se va a definir de nuevo el setpoint inicial para ver su comportamiento, se espera que esta vez la señal de salida siga al setpoint de forma aceptable, si es así, se tendría el comportamiento de la señal de control.

Es importante conocer la señal de control para poderla analizar, y de esta forma encontrar una ecuación capaz de adecuarla para que pueda ser inyectada en la planta, aunque también pudiera darse el caso que no es necesario adecuar la señal de control.

La Figura 7.2.7 muestra el setpoint frente a la salida del sistema.

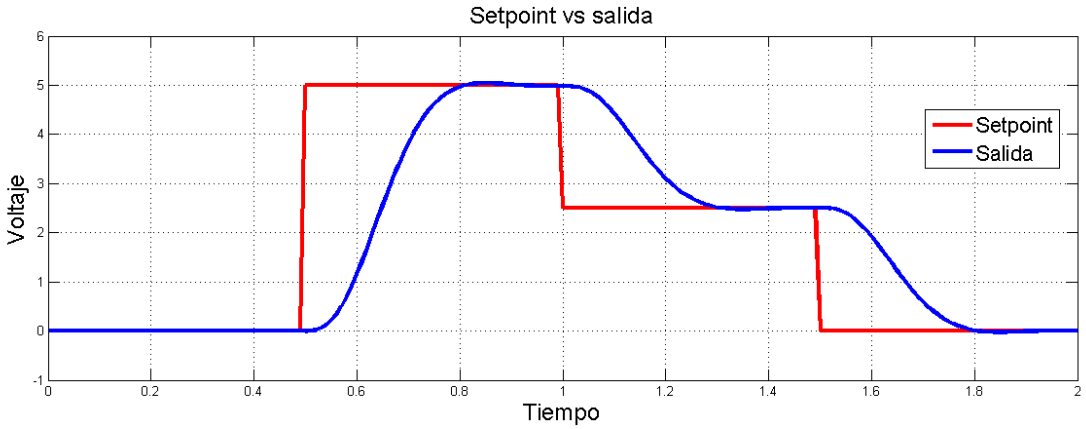


Figura 7.2.7: *Setpoint contra la salida.*

La Figura 7.2.8 muestra la señal de control

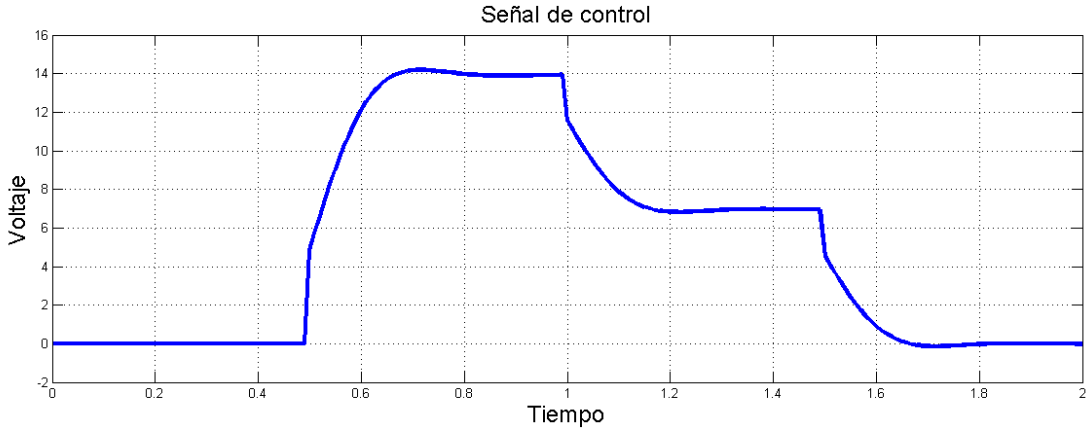


Figura 7.2.8*: Señal de control.*

La Figura 7.2.9 muestra la señal del error.

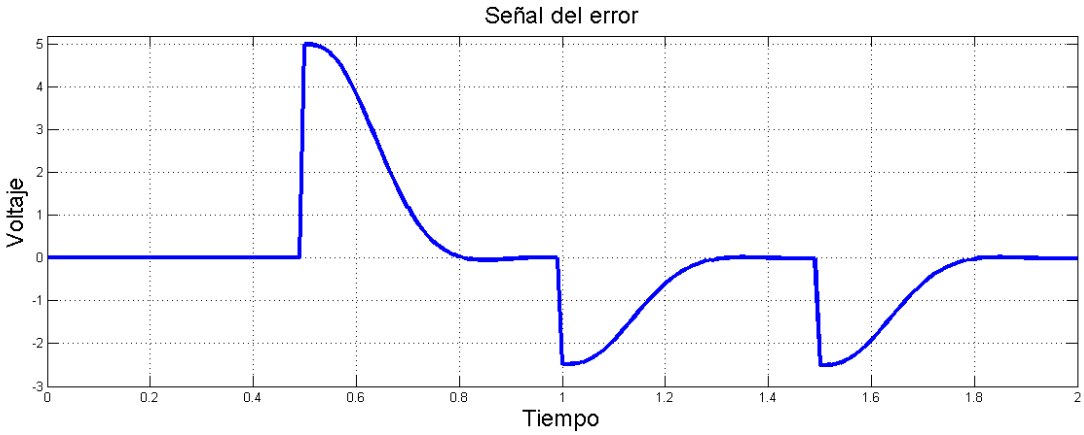


Figura 7.2.9: *Señal del error.*

Como se puede observar en la Figura 7.2.9 el sistema responde de forma aceptable, entiéndase por aceptable que su respuesta es mejor que la mostrada en la Figura 7.2.4, por lo tanto se procede a analizar la señal de control con el fin de adecuarla para ser utilizada por un puente H.

7.3 Análisis de la señal de control

De la señal de control se analizaron tres puntos de interés, el primero corresponde al cambio brusco que hace el motor al pasar de 0 grados a 180 grados, el segundo corresponde al cambio de 180 grados a 90 grados y por último de 90 grados a 0 grados, la Figura 7.3.1 muestra los datos arrojados por la señal de control en estos tres puntos.

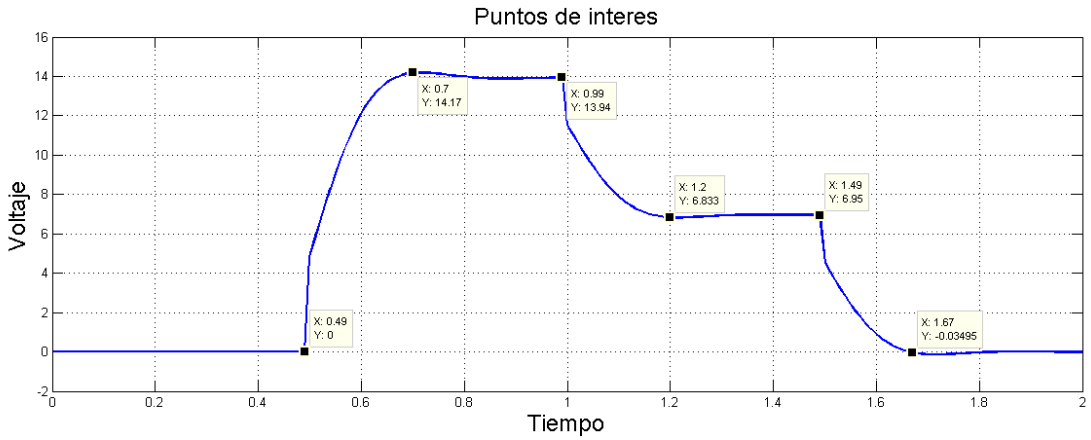


Figura 7.3.1: *Puntos de interés.*

El Cuadro 7.3.1 muestra los datos obtenidos

Posición [Grados]	Valor actual [Volts]	Valor final [Volts]
0-180	0	14.17
180-90	13.94	6.833
90-0	6.95	-0.0349

Cuadro 7.3.1: *Valores obtenidos.*

7.3.1 Aproximación I

De la tabla se puede extraer el valor máximo para adecuar la señal de control a los niveles de voltaje de entrada que van de 0 Volts a 5 Volts, se tendría que escalar la señal por un factor de 5/14.17.

La Figura 7.3.1.1 muestra el resultado de esta acción.

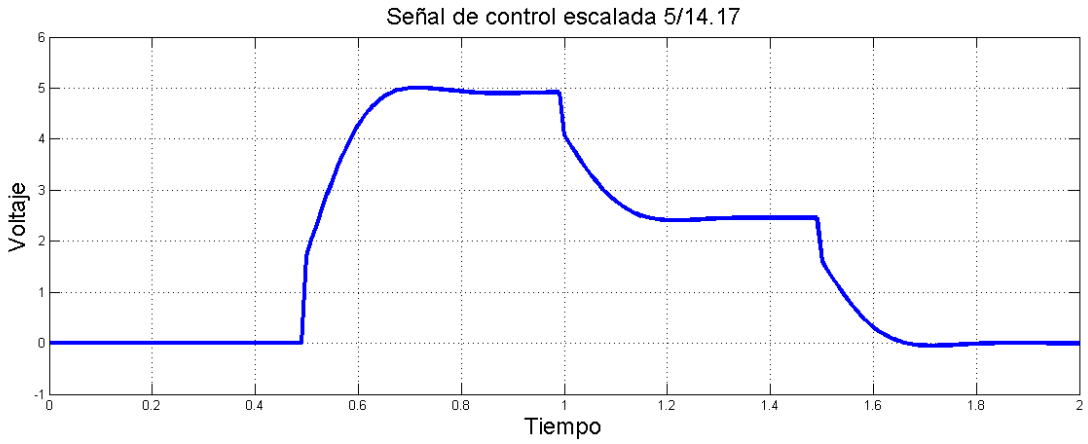


Figura 7.3.1.1: *Señal de control reducida por un factor de 5/14.17.*

Con esta reducción se espera encontrar los niveles adecuados para ser tratada, el Cuadro 7.3.1.1 muestra los nuevos valores obtenidos:

Posición [Grados]	Valor actual [Volts]	Valor final [Volts]
0-180	0	5.002
180-90	4.917	2.411
90-0	2.453	-0.02816

Cuadro 7.3.1.1: *Valores obtenidos.*

Con estos nuevos valores obtenidos se podría encontrar una relación lineal, si definimos 5Volts como un ciclo de trabajo del 100% entonces se tendría la siguiente relación:

$$DutyCicle = \frac{U_{Volt} * 100}{5}[\%]$$
$$U_{Volt} \rightarrow Voltaje\_señal\_control$$

El problema viene con el sentido de giro del motor, por lo tanto, esta relación solo es válida ya sea para giros positivos (0 grados a 180 grados) o negativos (180 grados a 0 grados), pero no para ambos, por lo tanto, solo sirve parcialmente.

7.3.2 Aproximación II

Una segunda aproximación serial calcular la razón de cambio de la señal de control.

La Figura 7.3.2.1 muestra la señal de control frente a la razón de cambio de la misma.

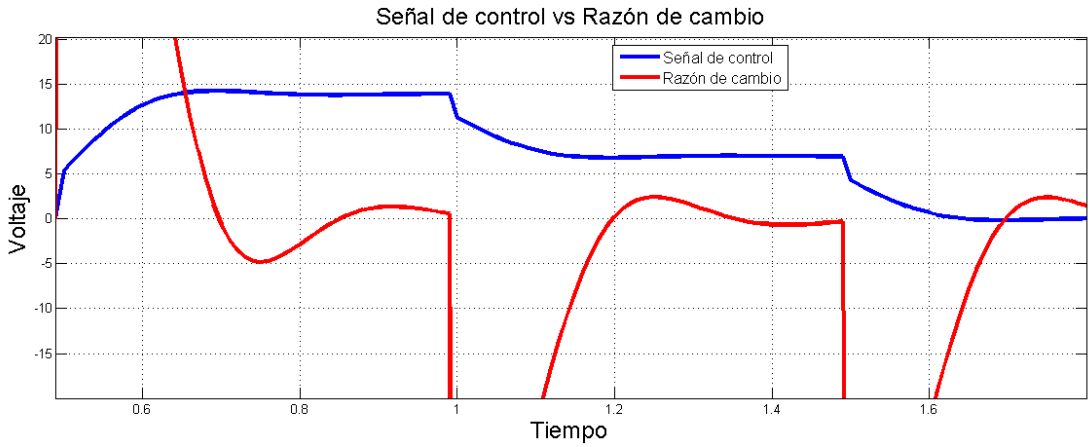


Figura 7.3.2.1: *Señal de control vs razón de cambio.*

La Figura 7.3.2.2 muestra la derivada de la señal de controla

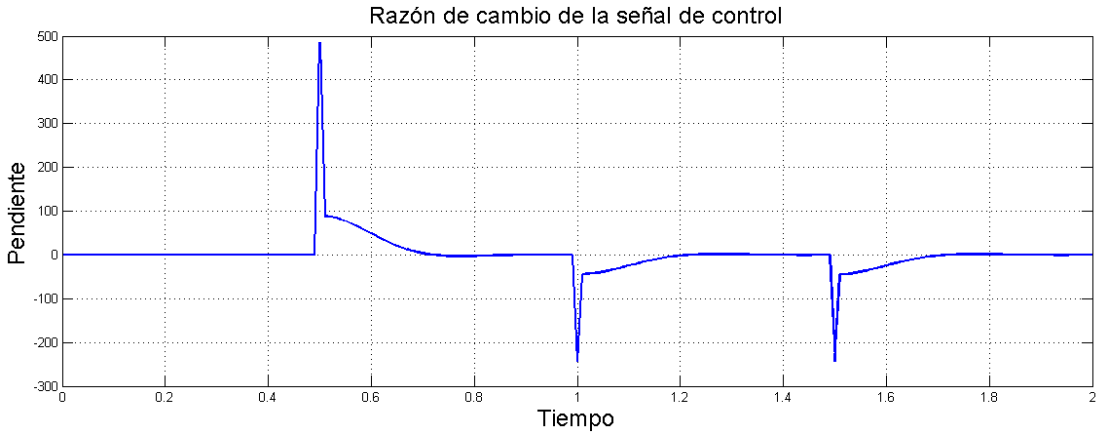


Figura 7.3.2.2: *Derivada de la señal de control.*

Analizando la razón de cambio de la señal de control, se obtienen pendientes que pueden corresponder al ciclo de trabajo de la señal PWM, lo mejor de todo es que no presenta el problema de la aproximación I, la cual solo es válida para un sentido de giro, en este caso el signo de la pendiente indica el sentido de giro, por lo tanto, la razón de cambio de la señal de control puede ser utilizada para controlar la posición del motor, en el Cuadro 7.3.2.1 se muestran los valores de las pendientes en puntos de interés:

Posición [Grados]	Signo	Rango
0-180	Positivo	485 a 0.02161
180-90	Negativo	-242.2 a 0.559
90-0	Negativo	-242.7 a 0.6269

Cuadro 7.3.2.1: *Valores obtenidos.*

7.4 Ecuación de la razón de cambio de la señal de control

La ecuación que define el cambio para la señal de control es la siguiente:

$$\frac{\Delta y}{\Delta x} = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

Como se sabe que el tiempo de muestreo es de 0.01ms entonces la ecuación se puede sobre escribir de la siguiente manera:

$$\frac{\Delta y}{\Delta x} = \frac{y_{k+1} - y_k}{0.01}$$

Ecuación M.N

Es importante señalar que es necesario definir limites superiores e inferiores, esto debido a los picos que presenta la señal, por ejemplo del Cuadro 7.3.2.1 se sabe que la pendiente de la señal cuando pasa de 0 grados a 180 grados es de 485, esto representa un ciclo de trabajo del 485%, por el contrario, cuando la señal pasa de 180 grados a 90 grados la pendiente es de -242 lo que equivale a un ciclo de trabajo del -242%, por lo tanto en software es necesario definir los límites para que cuando la pendiente sea mayor de 100 el ciclo de trabajo sea del 100% y cuando sea menor de 0 el ciclo de trabajo sea del 0%.

7.5 Control de giro del motor

De los resultados obtenidos del análisis de la señal de control, se optó por utilizar una señal PWM utilizando la técnica de signo magnitud, esta técnica permite controlar el motor mediante tres señales de control y un puente H, la primer señal corresponde al tiempo que permanece habilitado el motor, mientras que la segunda y tercera señal corresponde al sentido de giro del motor.

El modelo del circuito utilizado como puente H es L298N, la tabla de verdad del puente H es la siguiente:

Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

Tabla de verdad puente H L298N.

De la tabla de verdad se puede observar que el control de giro se establece mediante las entradas C y D las cuales corresponden a los pines *Input 1* e *Input 2* del circuito, mientras que la habilitación se hace mediante la entrada Ven el cual corresponde al pin *Enable* del circuito.

Por lo tanto para utilizar la técnica de signo magnitud para el puente H L298N, las conexiones quedarían de la siguiente manera, la salida PWM se conecta al pin *Enable*, mientras que los pines de dirección del puente H van conectados a cualquiera dos pin de propósito general del microcontrolador, se recomienda que se utilice un mismo puerto para tal fin, por ejemplo se puede utilizar la parte baja del puerto B, por lo tanto solo basta definir las dos posibles salidas, por ejemplo se puede definir 0x01 para la derecha y 0x02 para la izquierda, o viceversa, o en caso de que no se cuente con la cantidad de pines necesarios para el control de sentido de giro, se puede utilizar un inversor y solo un pin del microcontrolador.

El diagrama de conexión se muestra en la figura 7.5.1.

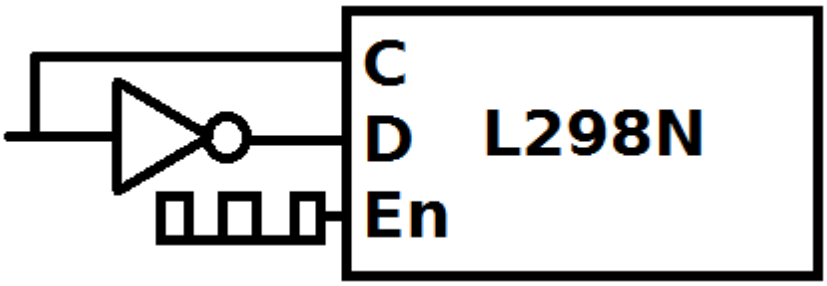


Figura 7.5.1: Diagrama de conexión.

7.6 Ecuaciones PWM

Las ecuaciones que rigen el comportamiento del módulo PWM del microcontrolador PIC16F1619 son las siguientes:

$$PWM\_Periodo = [(PR2 + 1] * 4 * T_{osc} (TMR2\_\Pr escaler)$$

$$T_{osc} = \frac{1}{F_{osc}}$$

$$Duty\_Cicle = \frac{CCPRxH : CCPRxL}{4(PR2 + 1)}$$

Ecuación periodo PWM ecuación ciclo de trabajo de la señal PWM en ambas ecuaciones, el valor de PR2 aparece, por lo tanto el primer paso es encontrar el valor correspondiente a este término.

Se eligió un periodo de 0.5 micro segundos equivalente a 200KHz, esta frecuencia no cumple ningún requerimiento en específico, se eligió enteramente para propósitos de prueba, de las tablas ofrecidas por el fabricante se tienen los siguientes valores:

Frecuencia PWM	200KHz
Prescalador Timer	1
Valor PR2	0x09
Máxima resolución (bits)	8

Una vez conocido el valor del parámetro PR2 la ecuación para el ciclo de trabajo queda de la siguiente manera:

$$Duty\_Cicle = \frac{CCPRxH : CCPRxL}{40}$$

Teniendo en cuenta que la resolución es de solo 8 bits, se puede elegir la parte alta o baja del registro CCPRx, se eligió utilizar la notación *little endian*, por lo tanto la ecuación final queda de la siguiente manera:

$$Duty\_Cicle = \frac{CCPRxL}{40}$$

De la ecuación anterior se puede concluir que para encontrar el ciclo de trabajo, es necesario multiplicarlo por 40, por lo tanto el registro CCPRxL tiene la siguiente relación:

$$CCPRxL = 40(Duty\_Cicle)$$

El registro CCPRxL tiene una longitud de 8 bits, esto quiere decir que el número máximo permitido en decimal es 255, pero teniendo en cuenta que un ciclo de trabajo del 100% corresponde a un valor de 40 (0x28 hex), este corresponde al número máximo permitido, valores mayores a 40 podrían dar ciclos de trabajos erróneos, por lo tanto es necesario modificar la ecuación de la razón de cambio de la señal de control para que caiga dentro de este rango de valores.

La ecuación de la derivada de la señal de control (Ecuación M.N) da el ciclo de trabajo en

decimales, por lo tanto va a ser necesario multiplicar esta ecuación por el tiempo de muestreo para que la salida esté dada en porcentajes, por lo tanto la ecuación para la razón de cambio queda de la siguiente forma:

$$\frac{\Delta y}{\Delta x} = y_{k+1} - y_k \qquad \ldots \textit{Ecuación 7}$$

La Figura 7.7.1 muestra la señal de salida de la derivada de la señal de control:

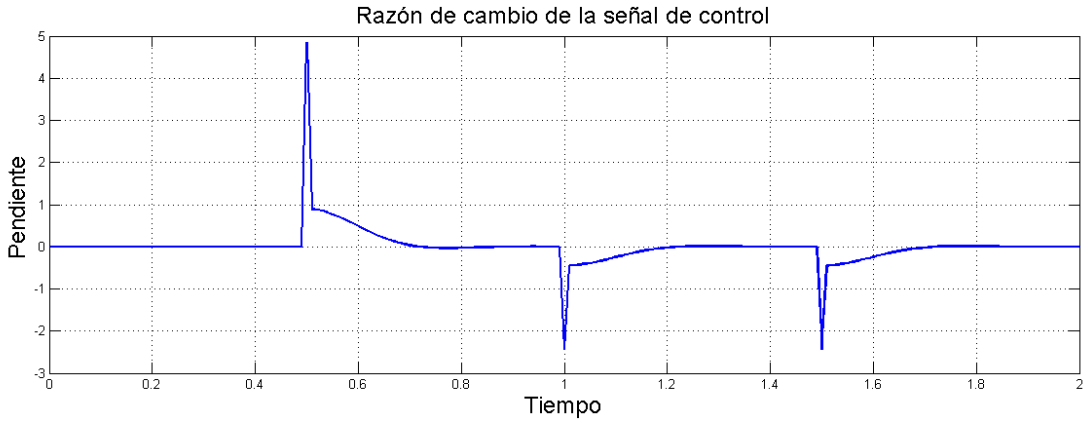


Figura 7.6.1: *Razón de cambio de la señal de control.*

Para ajustar la salida a los valores válidos requeridos por el registro CCPRxL es necesario multiplicar por 40, por lo tanto se tiene:

$$CCPRxL = 40(y_{k+1} - y_k)$$

La figura 7.6.2 muestra la salida de la relación anterior:

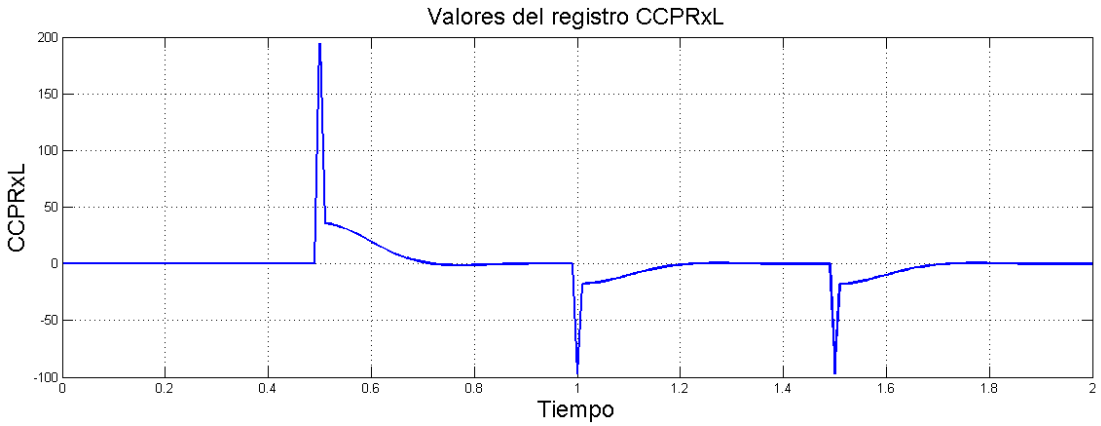


Figura 7.6.2: *Valores de los registros CCPRxL.*

El signo de la pendiente indica el sentido de giro, una vez establecido, dado que no existen ciclos de trabajo negativos, no tiene sentido la parte negativa, por lo tanto se aplica el valor absoluto a la salida del registro CCPRxL, por último es necesario definir el límite superior para que el valor de CCPRxL sea válido, en este caso el límite superior corresponde a 40, la Figura 7.6.3 muestra el resultado de la operación:

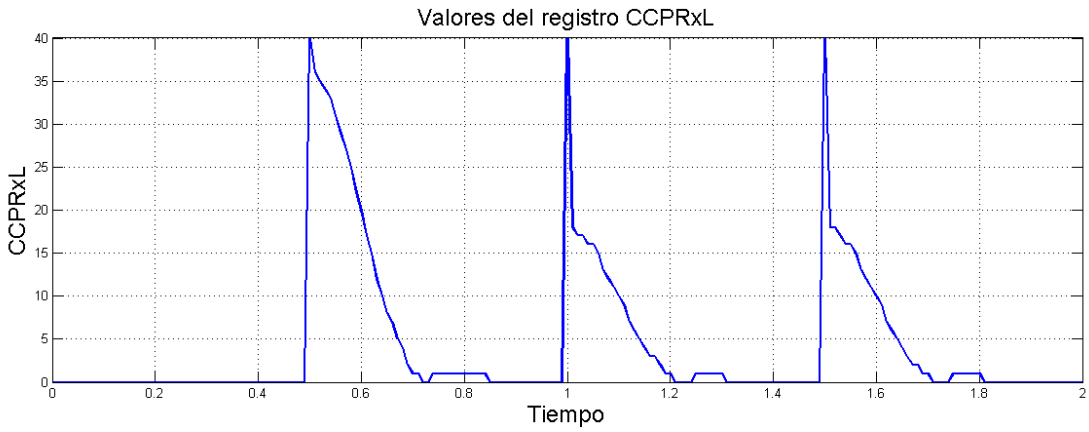


Figura 7.6.3: *Valores de los registros CCPRxL.*

De la Figura 7.6.3 se pueden apreciar pequeñas oscilaciones una vez que se alcanza el valor cero, esto es indicativo que se requieren ajustes finos a los parámetros Ki y Kp, pero eso va más allá del objetivo de esta tesis.

7.7 Implantación del controlador en el microcontrolador

El último paso fue programar el microcontrolador para implementar el control de posición de un motor de CD mediante un controlador PI.

Para evitar definir el tamaño del número fraccionario se utilizó una técnica de escalamiento.

Los registros del microcontrolador que almacenan las ganancias tienen una longitud de 16 bits, por lo tanto son viables para utilizar la técnica de escalamiento, esta técnica consiste en multiplicar el número fraccionario por un factor de escalamiento.

$$N_e = N_f * F_e$$
$$N_e \rightarrow Numero\_entero$$
$$N_f \rightarrow Numero\_fraccionario$$
$$F_e \rightarrow Factor\_de\_escalamiento$$

Para hacer los cálculos de las ganancias se utilizó un factor de escalamiento de 11 bits:

$$K_d = 0$$
$$K_i = (17.811992)(2^{11}) = 36478$$
$$K_p = (0.79190)(2^{11}) = 1621$$

Calculando los valores de K1,K2 y K3:

$$K_3 = \frac{K_d}{T_s} = 0$$
$$K_2 = -K_p - \frac{2K_d}{T_s} = -1621$$
$$K_1 = K_p + K_i T_s + \frac{K_d}{T_s} = 1621 + 364 = 1985$$

El bloque PWM en la Figura 7.7.1 engloba las ecuaciones para adaptar la salida del controlador, estas ecuaciones fueron calculadas en la sección *Ecuaciones PWM*. La Figura 7.7.1 muestra el diagrama de flujo del firmware.

Diagrama de flujo del firmware

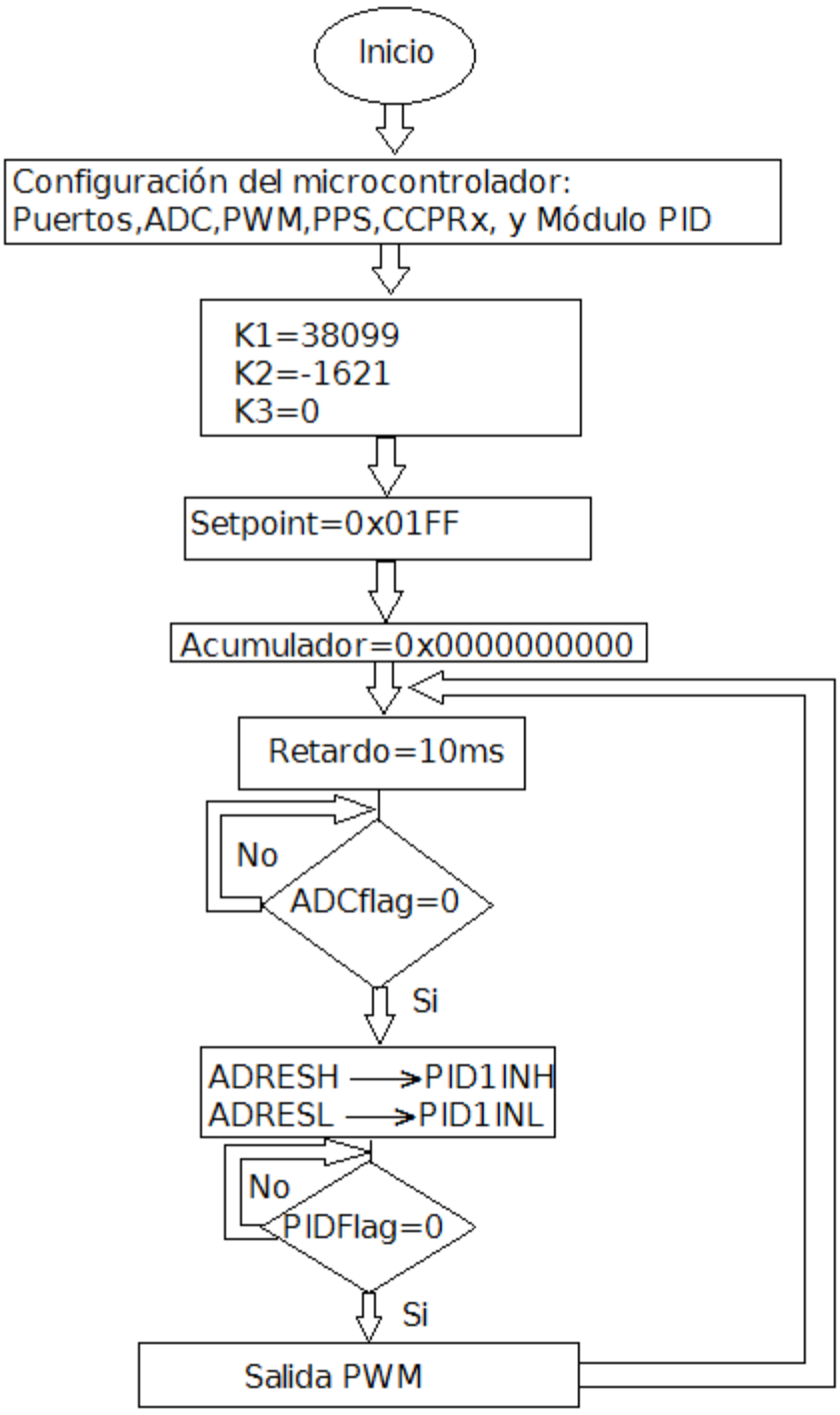


Figura 7.7.1: *Diagrama de flujo del fimrware para el control de posición.*

7.8 Resultados

Como se puede observar en el diagrama de flujo, gracias a la utilización de un bootloader la implementación del firmware para diseñar un controlador digital es directa, no se requiere utilizar ni definir los tipos de registros de las variables, tampoco es necesario diseñar un algoritmo para actualizar las ganancias, en pocas palabras, esto significa menos tiempo en el desarrollo del firmware y más tiempo en el desarrollo del controlador.

Sólo por mencionar la complejidad de implementar el algoritmo capaz de interpretar un número fraccionario, esto requiere definir el tamaño número de bits para el valor entero y el número de bits para la mantisa, dado que se trata de un número signado entonces el tamaño para la parte entera sería *n+1* mientras que para la parte fraccionaria quedaría en *n*, por otra parte, si se elige enviar los nuevos valores de las ganancias mediante comunicación serie, esto supone la conversión ascii a hexadecimal, esto se puede lograr fácilmente con una resta de 0x30 por cada número recibido por lo tanto para un numero fraccionario de 2 bytes para la parte entera y 3 bytes para la mantisa el tamaño del registro acumulador sería de 5 bytes dado que estamos trabajando en una arquitectura de 8 bits sería necesario concatenar 3 registros, ahora bien, si se quisiera tener una mejor precisión en los cálculos se tendría que aumentar el número de bytes tanto en la parte entera como en la parte fraccionaria lo que significa concatenar más registros de 8 bits, por último una vez decodificado el valor es necesario hacer los cálculos pertinentes para cada ganancia.

Cabe señalar que los valores obtenidos mediante simulink pudieran requerir de ajustes posteriores pero al menos sirven como base de partida para calibrar los parámetros, comparado con métodos basados en prueba y error donde no se tiene siquiera una referencia para empezar, por lo tanto el método presentado en esta tesis presenta ventajas.

El video que demuestra el funcionamiento del circuito se puede consultar en el siguiente enlace:

https://youtu.be/V6WQXnWxpVI

7.9 Conclusión

La utilización de un microcontrolador con un módulo especializado para implementar un controlador digital presenta ventajas frente a controladores basados en software, por parte del controlador basado en hardware solo es necesario esperar las banderas del módulo para continuar con los cálculos (esto en caso de que no se haga uso de la interrupción), mientras que en software se requiere mantener al procesador calculando operaciones, dicho de otra forma, implementado en hardware requiere menos ciclos de reloj lo que significa menos energía y sobre todo, esto supone una fuerte reducción en tiempo de ejecución para una sola tarea.

Por otra parte la utilización de un bootloader, en esta caso JBoot, facilita mucho el tiempo de desarrollo del firmware y puesta en marcha del controlador, esto es debido a que en muy pocas ocasiones las ganancias del controlador son números enteros, por lo general se tratan de números fraccionarios signados, por tanto para actualizar los valores de las ganancias es necesario implementar en software un algoritmo capaz de interpretar los valores, otra de las ventajas que presenta la utilización de un bootloader recae en el diseño del PCB, es un hecho que entre menos pistas contenga un esquema es más probable que el software de enrutamiento encuentre una solución, dicho esto, la utilización del bootloader elimina las pistas utilizadas para programar el microcontrolador mediante ICSP, esto en la práctica pudiera significar un PCB de una sola cara o un PCB de doble cara lo cual significa tiempo y gastos adicional de producción.

Bibliografía

Libros

[1]Angulo, José Ma. (2003). Microcontroladores PIC. La Clave del Diseño. España. Thomson.

[2]Angulo, José Ma. (2006). Microcontroladores PIC. Diseño Práctico de aplicaciones. España. Mc Graw-Hill. Tercera edición.

[3]Malvino, Albert. (2007). Principios de Electrónica. España. Mc Graw Hill. 7ma Edición.

[4]Palacios, Enrique. (2004). Microcontrolador PIC16F84 Desarrollo de Proyectos. México. Alfaomega Ra-Ma. Primera Edición.

[5]Peñaloza, Ernesto, (2004). Fundamentos de Programación C/C++, México. Alfaomega, Tercera Edición.

Manuales

[1]Agilent DSO5032A 300 MHz 2 ch Oscilloscope

[2]Translator Example. Reston Condit. Microchip Technology Inc. 2004.

[3] Calculating Program Memory Checksums Using a PIC16F87X. Rodger Richey. Microchip Technology Inc. 1998.

[4]Downloading HEX Files to PIC16F87X PICmicro® Microcontrollers. Rodger Richey. Microchip Technology Inc. 1998.

[5]Low Cost USB Microcontroller Programmer. The building of the PICkitTM 1

[6]FLASH Starter Kit. Reston Condit. Dan Butler. Microchip Technology Inc. 2003.

[7]Generating High Voltage Using the PIC16C781/782. Ross Fosler. Microchip Technology Inc. 2005.

[8]In-Circuit Serial ProgrammingTM(ICSPTM) Guide. Microchip Technology Inc. 2003.

[9]Midrange and Enhanced Reference Manuals (Chapter 2. Oscillator). Microchip Technology Inc.

[10]Midrange and Enhanced Reference Manuals (Chapter 3. Reset). Microchip Technology Inc.

Sitios Web

[1]Control Tutorials for Matlab & Simulink, http://ctms.engin.umich.edu/CTMS/index.php?example=MotorPosition&section=SystemModeling. Consultado 22 de febrero de 2016

[2]Hexadecimal Object File Format Specification, Revision A http://microsym.com/editor/assets/intelhex.pdf Consultado 2 de marzo de 2016

[3] PIC16F87X Bootloader With PIC Flash Module http://www.mcjournal.com/articles/arc103/arc103.htm Consultado 2 Marzo de 2016

[4]Jogu Bootloader github repository

https://github.com/jogu/pic-bootloaders Consultado 16 de Marzo de 2016

[5]PIC16f877 Monitor and Bootloader

http://public.wsu.edu/~jackdoll/jmon/ Consultado 28 Marzo 2016

[6]Frank Sergeant's Pikme (PIC) Programming Page

http://pygmy.utoh.org/pikme/index.html Consultado 30 Marzo de 2016

Apéndice A:

Código fuente JBoot

```
MPASM  5.51                   JBOOT1827.ASM   4-17-2016  15:40:15         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      00001 ;*******************************************
                      00002 ;E. David Rojas Serrano <edx@edx-twin3.org>          *
                      00003 ;http://www.edx-twin3.org                           *
                      00004 ;Codigo JBoot para el PIC16F1827                    *
                      00005 ;*******************************************
                      00006  list p = PIC16F1827
                      00007  errorlevel -302
                      00008  errorlevel -303
                      00009  errorlevel -307
                      00010
8007   0FC2           00011  __CONFIG 0x8007, 0xFFFA & 0xFFE7 & 0xFFDF & 0xFFFF
& 0xFFFF & 0xFFFF & 0xFFFF & 0xFFFF & 0xEFFF & 0xDFF
                         F
8008   18FF           00012  __CONFIG 0x8008, 0xFFFF & 0xFEFF & 0xFDFF & 0xFBFF
& 0xDFFF
                      00013
0000                  00014  org 0x00
0000   318F 2F00      00015  lgoto _JBoot
0002                  00016  org 0x02
0002   2802           00017 _BdeProg goto $+0
                      00018
0F00                  00019  org 0xF00
0F00                  00020 _JBoot
0F00   0021           00021  movlb 0x01
0F01   3068           00022  movlw 0x68
0F02   0099           00023  movwf 0x99
0F03   1E9A           00024  btfss 0x9A,5
0F04   2F03           00025  goto $-1
0F05   3002           00026  movlw 0x02
0F06   008D           00027  movwf 0x8D
0F07   3087           00028  movlw 0x87
0F08   0095           00029  movwf 0x95
0F09   0023           00030  movlb 0x03
0F0A   018D           00031  clrf 0x18D
0F0B   3081           00032  movlw .129
0F0C   009B           00033  movwf 0x19B
0F0D   179D           00034  bsf 0x19D,7
```

```
0F0E   161D          00035  bsf 0x19D,4
0F0F   169E          00036  bsf 0x19E,5
0F10   151E          00037  bsf 0x19E,2
0F11   0020          00038  movlb 0x00
0F12   304D          00039  movlw .77
0F13   00A0          00040  movwf 0x20
0F14   0195          00041  clrf 0x15
0F15   1A91          00042 _TLoop btfsc 0x11,5
0F16   2F1D          00043  goto _HostReply
0F17   1D0B          00044  btfss 0x0B,2
0F18   2F15          00045  goto $-3
0F19   110B          00046  bcf 0x0B,2
0F1A   0BA0          00047  decfsz 0x20,1
0F1B   2F15          00048  goto _TLoop
0F1C   2FA0          00049  goto _MNormal
0F1D   0023          00050 _HostReply movlb 0x03
0F1E   0819          00051  movf 0x199,0
0F1F   00A0          00052  movwf 0x1A0
```



```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0F20   3055          00053  movlw 0x55
0F21   0620          00054  xorwf 0x1A0,0
0F22   1D03          00055  btfss 0x03,2
0F23   2FA0          00056  goto _MNormal
0F24   304A          00057  movlw 0x4A
0F25   279C          00058  call _EnvDato
0F26   3078          00059  movlw .120
0F27   00A0          00060  movwf 0x1A0
0F28   0192          00061  clrf 0x192
0F29   0191          00062  clrf 0x191
0F2A   1795          00063  bsf 0x195,7
0F2B   1315          00064  bcf 0x195,6
0F2C   1615          00065 _BorBlock bsf 0x195,4
0F2D   1515          00066  bsf 0x195,2
0F2E   2789          00067  call _secREQ
0F2F   1095          00068  bcf 0x195,1
0F30   1A15          00069  btfsc 0x0195,4
0F31   2F30          00070  goto $-1
0F32   3020          00071  movlw 0x20
0F33   0791          00072  addwf 191+0,1
0F34   1D03          00073  btfss 0x03,2
0F35   2F37          00074  goto $+2
```

```
0F36   0A92          00075  incf 0x191+1,1
0F37   0BA0          00076  decfsz 0x1A0,1
0F38   2F2C          00077  goto _BorBlock
0F39   1515          00078  bsf 0x195,2
0F3A   1695          00079  bsf 0x195,5
0F3B   0192          00080  clrf 0x192
0F3C   0191          00081  clrf 0x191
0F3D   3031          00082  movlw 0x31
0F3E   0094          00083  movwf 0x194
0F3F   308F          00084  movlw 0x8F
0F40   0093          00085  movwf 0x193
0F41   2789          00086  call _secREQ
0F42   0A91          00087  incf 0x191,1
0F43   302F          00088  movlw 0x2F
0F44   0094          00089  movwf 0x194
0F45   3000          00090  movlw 0x00
0F46   0093          00091  movwf 0x193
0F47   2789          00092  call _secREQ
0F48   1295          00093  bcf 0x195,5
0F49   2789          00094  call _secREQ
0F4A   1115          00095  bcf 0x195,2
0F4B   0186          00096  clrf 0x06
0F4C   1515          00097 _SecProg bsf 0x195,2
0F4D   1695          00098  bsf 0x195,5
0F4E   2791          00099  call _RecepDatos
0F4F   00A1          00100  movwf 0x1A1
0F50   0D89          00101  rlf 0x09,1
0F51   2797          00102  call _AddCHKSUM
0F52   2791          00103  call _RecepDatos
0F53   0092          00104  movwf 0x192
0F54   0085          00105  movwf 0x05
```



```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0F55   2791          00106  call _RecepDatos
0F56   0091          00107  movwf 0x191
0F57   0084          00108  movwf 0x04
0F58   0805          00109  movf 0x05,0
0F59   0785          00110  addwf 0x05,1
0F5A   0804          00111  movf 0x04,0
0F5B   0784          00112  addwf 0x04,1
0F5C   1C03          00113  btfss 0x03,0
0F5D   2F5F          00114  goto $+2
```

```
0F5E   0A85          00115  incf 0x04+1,1
0F5F   0805          00116  movf 0x05,0
0F60   2797          00117  call _AddCHKSUM
0F61   0804          00118  movf 0x04,0
0F62   2797          00119  call _AddCHKSUM
0F63   2791          00120 _loopDRCN call _RecepDatos
0F64   0094          00121  movwf 0x194 ;EEDATH
0F65   2797          00122  call _AddCHKSUM
0F66   2791          00123  call _RecepDatos
0F67   0093          00124  movwf 0x193 ;EEDATL
0F68   2797          00125  call _AddCHKSUM
0F69   2789          00126  call _secREQ
0F6A   0BA1          00127  decfsz 0x1A1,1
0F6B   2F80          00128  goto _IncDir
0F6C   0806          00129  movf 0x06,0
0F6D   0989          00130  comf 0x09,1
0F6E   3E01          00131  addlw 0x01
0F6F   00A2          00132  movwf 0x1A2
0F70   0184          00133  clrf 0x04
0F71   0186          00134  clrf 0x06
0F72   0020          00135  movlb 0x00
0F73   1E91          00136  btfss 0x11,5
0F74   2F73          00137  goto $-1
0F75   0023          00138  movlb 0x03
0F76   0819          00139  movf 0x199,0
0F77   0622          00140  xorwf 0x1A2,0
0F78   1D03          00141  btfss 0x03,2
0F79   2F7D          00142  goto _CHKInC
0F7A   304B          00143  movlw 0x4B
0F7B   279C          00144  call _EnvDato
0F7C   2F85          00145  goto _SecGrab
0F7D   3045          00146 _CHKInC movlw 0x45 ;E
0F7E   279C          00147  call _EnvDato
0F7F   2F4C          00148  goto _SecProg
0F80   0A91          00149 _IncDir incf 0x191+0,1
0F81   1D03          00150  btfss 0x03,2
0F82   2F84          00151  goto $+2
0F83   0A92          00152  incf 0x191+1,1
0F84   2F63          00153  goto _loopDRCN
0F85   1295          00154 _SecGrab bcf 0x195,5
0F86   2789          00155  call _secREQ
0F87   1115          00156  bcf 0x195,2
0F88   2F4C          00157  goto _SecProg
0F89   3055          00158 _secREQ movlw 0x55
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0F8A  0096          00159  movwf 0x196
0F8B  30AA          00160  movlw 0xAA
0F8C  0096          00161  movwf 0x196
0F8D  1495          00162  bsf 0x195,1
0F8E  0000          00163  nop
0F8F  0000          00164  nop
0F90  0008          00165  return
0F91  0020          00166 _RecepDatos movlb 0x00
0F92  1E91          00167  btfss 0x11,5
0F93  2F92          00168  goto $-1
0F94  0023          00169  movlb 0x03
0F95  0819          00170  movf 0x199,0
0F96  0008          00171  return
0F97  0786          00172 _AddCHKSUM addwf 0x06,1
0F98  1C03          00173  btfss 0x03,0
0F99  2F9B          00174  goto $+2
0F9A  0A87          00175  incf 0x06+1,1
0F9B  0008          00176  return
0F9C  009A          00177 _EnvDato movwf 0x19A
0F9D  1C9E          00178  btfss 0x19E,1
0F9E  2F9D          00179  goto $-1
0F9F  0008          00180  return
0FA0  0023          00181 _MNormal movlb 0x03
0FA1  139D          00182  bcf 0x19D,7
0FA2  0020          00183  movlb 0x00
0FA3  3180 2802      00184  lgoto _BdeProg
                    00185  end

MPASM  5.51                JBOOT1827.ASM   4-17-2016  15:40:15        PAGE  5

SYMBOL TABLE
  LABEL                     VALUE

_AddCHKSUM                  00000F97
_BdeProg                    00000002
_BorBlock                   00000F2C
_CHKInC                     00000F7D
_EnvDato                    00000F9C
_HostReply                  00000F1D
_IncDir                     00000F80
_JBoot                      00000F00
_MNormal                    00000FA0
_RecepDatos                 00000F91
```

```
_SecGrab                          00000F85
_SecProg                          00000F4C
_TLoop                            00000F15
__16F1827                         00000001
_loopDRCN                         00000F63
_secREQ                           00000F89


MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXX---------- ---------- ---------- ----------
0F00 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0F40 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0F80 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXX---------- --------
--------
8000 : -------XX------- ---------- ---------- ----------

All other memory blocks unused.

Program Memory Words Used:   170
Program Memory Words Free:  3926


Errors   :         0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,    64 suppressed
```

Apéndice C:

C1.1

Código fuente pantalla led

```
MPASM  5.51                   MATRIZ.ASM   4-30-2016  20:56:04         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      00001 ;*******************************************
                      00002 ;E. David Rojas Serrano <edx@edx-twin3.org
                      00003 ;http://www.edx-twin3.org
                      00004 ;*******************************************
                      00005 ;Pantalla: PD 4437
                      00006 ;RB7-RB0 -> D7-D0
```

```
                00007 ;RA0-RA2 -> Address input
                00008 ;RA3 -> Reset
                00009 ;R4 -> WR
                00010 ;R5 -> RD
                00011 ;***********************************************



                  00014  errorlevel -302
0000              00015  org 0x00
0000  158A          00016  bsf 0x0A,3
0001  160A          00017  bsf 0x0A,4
0002  2F00          00018  goto 0x700
                00019 ;***************************************
0003  0183          00020  clrf 0x03 ;Banco 0
                00021 ;***********************************************
                00022 ;Configurando Puerto
                00023 ;***********************************************
0004  1683          00024  bsf 0x03,5 ;Banco 1
0005  0185          00025  clrf 0x85 ;TRISA
0006  0186          00026  clrf 0x86 ;TRISB
                00027 ;Configuracion del reloj
0007  3087          00028  movlw 0x87 ;Tiempo base 65.
0008  0081          00029  movwf 0x81 ;OPTION_REG
0009  1703          00030  bsf 0x03,6 ;Banco 3
000A  0188          00031  clrf 0x188 ;ANSEL
000B  0189          00032  clrf 0x189 ;ANSELH
000C  0183          00033  clrf 0x03 ;Banco 0
                00034 ;***********************************************
                00035 ;Test Matriz
                00036 ;***********************************************
                00037  ;clrf 0x05 ;Control Word
                00038  ;bsf 0x05,3 ;RESET off
                00039  ;movlw 0x43 ;TEST
                00040  ;movwf 0x06 ;PORTB
                00041  ;goto $+0
                00042 ;***********************************************
                00043 ;Configurando Matriz
                00044 ;***********************************************
                00045 ; A2 A1 A0
                00046 ; 0  X  X Control Word
                00047 ;***********************************************
                00048 ;Control Word
                00049 ;***********************************************
                00050 ; D7 D6 D5 D4 D3 D2 D1 D0
                00051 ; 0  0  X  X  X  X  0  0  BLANK
                00052 ; 0  0  X  X  X  X  1  1  FULL BRIGHTNESS
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00053 ;*************************************************
000D  3010           00054  movlw 0x10 ;RD = Enable, Wr = Disable
000E  0085           00055  movwf 0x05 ;PORTA
000F  3028           00056  movlw 0x28 ;RESET off
0010  0085           00057  movwf 0x05 ;PORTA
0011  3003           00058  movlw 0x03 ;Full Brightness
0012  0086           00059  movwf 0x06 ;PORTB
                     00060 ;*************************************************
                     00061 ;Variables
                     00062 ;0x20 -> Contador de dirección
                     00063 ;0x21 -> Timer
                     00064 ;0x22 -> Index tabla
                     00065 ;0x23 -> Numero total de caracteres
                     00066 ;*************************************************
0013  302F           00067  movlw 0x2F ;Inicio a la derecha
0014  00A1           00068  movwf 0x21 ;Registo de direccion
0015  01A2           00069  clrf 0x22 ;Registro index FSR
0016  3004           00070  movlw 0x04 ;Numero total de caracteres
0017  00A3           00071  movwf 0x23
                     00072 ;*************************************************
                     00073 ;Mensaje
                     00074 ;Direccion de inicio 0x24 - 0x7F
                     00075 ;*************************************************
0018  3024           00076   movlw 0x24 ;Direccion de inicio de la tabla
0019  0084           00077   movwf 0x04 ;FSR
001A  3001           00078   movlw HIGH(_mensaje)
001B  008A           00079   movwf 0x0A
001C  0822           00080   movf 0x22,0
001D  2100           00081   call _mensaje
001E  0080           00082   movwf 0x00 ;INDF
001F  0AA2           00083   incf 0x22,1
0020  0A84           00084   incf 0x04,1 ;FSR+1
0021  0BA3           00085   decfsz 0x23,1
0022  281C           00086   goto $-6
0023  018A           00087   clrf 0x0A ;Volviendo a poner el PCLATH en 0
                     00088 ;*************************************************
                     00089 ;Estableciendo vectores de inicio FSR
                     00090 ;*************************************************
0024  3024           00091  movlw 0x24
0025  0084           00092  movwf 0x04
```

```
                  00093 ;*************************************************
                  00094 ;*************************************************
                  00095 ;Estableciendo numero total de caracteres
                  00096 ;*************************************************
0026   3004       00097  movlw 0x04
0027   00A3       00098  movwf 0x23
                  00099 ;*************************************************
                  00100 ;Programa Principal
                  00101 ;*************************************************
0028   2030       00102 _m1 call _direccion
0029   0800       00103  movf 0x00,0 ;INDF
002A   0086       00104  movwf 0x06 ;PORTB
002B   203D       00105  call _lectura

MPASM  5.51                MATRIZ.ASM   4-30-2016  20:56:04        PAGE  3


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE
002C   0A84       00107  incf 0x04,1
002D   0BA3       00108  decfsz 0x23,1
002E   2828       00109  goto _m1
                  00110 ;*************************************************
                  00111 ;Estableciendo vectores de inicio FSR
                  00112 ;*************************************************
                  00113  ;movf 0x04,0 ;Leyendo posicion actual del FSR
                  00114 ;*************************************************
                  00115 ;*************************************************
                  00116 ;Estableciendo numero total de caracteres
                  00117 ;*************************************************
                  00118  ;movlw 0x04
                  00119  ;movwf 0x23
002F   282F       00120  goto $+0
                  00121 ;*************************************************
                  00122 ;Algoritmo de direccion
                  00123 ;*************************************************
                  00124 ;Posición del caracter
                  00125 ;*************************************************
                  00126 ; A2 A1 A0
                  00127 ; 1  0  0 -> Digit 0 (Derecha)
                  00128 ; 1  0  1 -> Digit 1
                  00129 ; 1  1  0 -> Digit 2
                  00130 ; 1  1  1 -> Digit 3 (Izquierda)
                  00131 ;*************************************************
0030   0821       00132 _direccion movf 0x21,0 ;Cargando direccion
0031   0085       00133  movwf 0x05 ;PORTA
0032   03A1       00134  decf 0x21,1
```

```
0033   302B          00135  movlw 0x2B ;Direcion 2C - 1 para que funcione
0034   0621          00136  xorwf 0x21,0
0035   1D03          00137  btfss 0x03,2
0036   0008          00138  return
0037   302F          00139  movlw 0x2F ;Empieza a la derecha
0038   00A1          00140  movwf 0x21
0039   0008          00141  return
                     00148 ;*************************************************
                     00149 ;Lectura
                     00150 ;*************************************************
003D   1605          00151 _lectura bsf 0x05,4 ;WR = 0
003E   1285          00152  bcf 0x05,5 ;RD = 1
003F   0008          00153  return

MPASM  5.51                   MATRIZ.ASM   4-30-2016  20:56:04        PAGE  4


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00166 ;*************************************************
                     00167 ;Mensaje
                     00168 ;*************************************************
0100                 00169  org 0x100
0100   0782          00170 _mensaje addwf 0x02,1
0101   3455 3441 3443 00171  dt "UACM"
       344D
                     00172  end
MPASM  5.51                   MATRIZ.ASM   4-30-2016  20:56:04        PAGE  5


SYMBOL TABLE
  LABEL                           VALUE

__16F886                          00000001
_antilectura                      0000003A
_direccion                        00000030
_lectura                          0000003D
_m1                               00000028
_mensaje                          00000100


MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0100 : XXXXX---------- ---------------- ---------------- ----------------
```

All other memory blocks unused.

Program Memory Words Used: 69
Program Memory Words Free: 8123

Errors   : 0
Warnings : 0 reported, 0 suppressed
Messages : 0 reported, 5 suppressed

C1.2

Código fuente pantalla led

MPASM 5.51 MATRIZ.ASM 4-30-2016 21:02:53 PAGE 1

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001 ;*************************************************
                     00002 ;E. David Rojas Serrano <edx@edx-twin3.org
                     00003 ;http://www.edx-twin3.org
                     00004 ;*************************************************
                     00005 ;Pantalla: PD 4437
                     00006 ;RB7-RB0 -> D7-D0
                     00007 ;RA0-RA2 -> Address input
                     00008 ;RA3 -> Reset
                     00009 ;R4 -> WR
                     00010 ;R5 -> RD
                     00011 ;*************************************************

                     00014  errorlevel -302
0000                 00015  org 0x00
0000   158A          00016  bsf 0x0A,3
0001   160A          00017  bsf 0x0A,4
0002   2F00          00018  goto 0x700
```

```
                  00019 ;*****************************************
0003   0183       00020  clrf 0x03 ;Banco 0
                  00021 ;*************************************************
                  00022 ;Configurando Puerto
                  00023 ;*************************************************
0004   1683       00024  bsf 0x03,5 ;Banco 1
0005   0185       00025  clrf 0x85 ;TRISA
0006   0186       00026  clrf 0x86 ;TRISB
                  00027 ;Configuracion del reloj
0007   3087       00028  movlw 0x87 ;Tiempo base 65.
0008   0081       00029  movwf 0x81 ;OPTION_REG
0009   1703       00030  bsf 0x03,6 ;Banco 3
000A   0188       00031  clrf 0x188 ;ANSEL
000B   0189       00032  clrf 0x189 ;ANSELH
000C   0183       00033  clrf 0x03 ;Banco 0
                  00034 ;*************************************************
                  00035 ;Test Matriz
                  00036 ;*************************************************
                  00037  ;clrf 0x05 ;Control Word
                  00038  ;bsf 0x05,3 ;RESET off
                  00039  ;movlw 0x43 ;TEST
                  00040  ;movwf 0x06 ;PORTB
                  00041  ;goto $+0
                  00042 ;*************************************************
                  00043 ;Configurando Matriz
                  00044 ;*************************************************
                  00045 ; A2 A1 A0
                  00046 ; 0  X  X Control Word
                  00047 ;*************************************************
                  00048 ;Control Word
                  00049 ;*************************************************
                  00050 ; D7 D6 D5 D4 D3 D2 D1 D0
                  00051 ; 0  0  X  X  X  X  0  0  BLANK
                  00052 ; 0  0  X  X  X  X  1  1  FULL BRIGHTNESS

MPASM  5.51                MATRIZ.ASM   4-30-2016  21:02:53        PAGE  2


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                  00053 ;*************************************************
000D   3010       00054  movlw 0x10 ;RD = Enable, Wr = Disable
000E   0085       00055  movwf 0x05 ;PORTA
000F   3028       00056  movlw 0x28 ;RESET off
0010   0085       00057  movwf 0x05 ;PORTA
0011   3003       00058  movlw 0x03 ;Full Brightness
```

```
0012  0086          00059  movwf 0x06 ;PORTB
                    00060 ;*********************************************
                    00061 ;Variables
                    00062 ;0x20 -> Contador de dirección
                    00063 ;0x21 -> Timer
                    00064 ;0x22 -> Index tabla
                    00065 ;0x23 -> Numero total de caracteres
                    00066 ;*********************************************
0013  302F          00067  movlw 0x2F ;Inicio a la derecha
0014  00A1          00068  movwf 0x21 ;Registo de direccion
0015  01A2          00069  clrf 0x22 ;Registro index FSR
0016  3004          00070  movlw 0x04 ;Numero total de caracteres
0017  00A3          00071  movwf 0x23
                    00072 ;*********************************************
                    00073 ;Mensaje
                    00074 ;Direccion de inicio 0x24 - 0x7F
                    00075 ;*********************************************
0018  3024          00076   movlw 0x24 ;Direccion de inicio de la tabla
0019  0084          00077   movwf 0x04 ;FSR
001A  3001          00078   movlw HIGH(_mensaje)
001B  008A          00079   movwf 0x0A
001C  0822          00080   movf 0x22,0
001D  2100          00081   call _mensaje
001E  0080          00082   movwf 0x00 ;INDF
001F  0AA2          00083   incf 0x22,1
0020  0A84          00084   incf 0x04,1 ;FSR+1
0021  0BA3          00085   decfsz 0x23,1
0022  281C          00086   goto $-6
0023  018A          00087   clrf 0x0A ;Volviendo a poner el PCLATH en 0
                    00088 ;*********************************************
                    00089 ;Estableciendo vectores de inicio FSR
                    00090 ;*********************************************
0024  3024          00091  movlw 0x24
0025  0084          00092  movwf 0x04
                    00093 ;*********************************************
                    00094 ;*********************************************
                    00095 ;Estableciendo numero total de caracteres
                    00096 ;*********************************************
0026  3004          00097  movlw 0x04
0027  00A3          00098  movwf 0x23
                    00099 ;*********************************************
                    00100 ;Programa Principal
                    00101 ;*********************************************
0028  2031          00102 _m1 call _direccion
0029  0800          00103  movf 0x00,0 ;INDF
002A  0086          00104  movwf 0x06 ;PORTB
002B  203E          00105  call _lectura
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

002C  2041          00106  call _timer
002D  0A84          00107  incf 0x04,1
002E  0BA3          00108  decfsz 0x23,1
002F  2828          00109  goto _m1
                    00110 ;*********************************************
                    00111 ;Estableciendo vectores de inicio FSR
                    00112 ;*********************************************
                    00113  ;movf 0x04,0 ;Leyendo posicion actual del FSR
                    00114 ;*********************************************
                    00115 ;*********************************************
                    00116 ;Estableciendo numero total de caracteres
                    00117 ;*********************************************
                    00118  ;movlw 0x04
                    00119  ;movwf 0x23
0030  2830          00120  goto $+0
                    00121 ;*********************************************
                    00122 ;Algoritmo de direccion
                    00123 ;*********************************************
                    00124 ;Posición del caracter
                    00125 ;*********************************************
                    00126 ; A2 A1 A0
                    00127 ; 1  0  0 -> Digit 0 (Derecha)
                    00128 ; 1  0  1 -> Digit 1
                    00129 ; 1  1  0 -> Digit 2
                    00130 ; 1  1  1 -> Digit 3 (Izquierda)
                    00131 ;*********************************************
0031  0821          00132 _direccion movf 0x21,0 ;Cargando direccion
0032  0085          00133  movwf 0x05 ;PORTA
0033  03A1          00134  decf 0x21,1
0034  302B          00135  movlw 0x2B ;Direcion 2C - 1 para que funcione
0035  0621          00136  xorwf 0x21,0
0036  1D03          00137  btfss 0x03,2
0037  0008          00138  return
0038  302F          00139  movlw 0x2F ;Empieza a la derecha
0039  00A1          00140  movwf 0x21
003A  0008          00141  return
                    00142 ;*********************************************
                    00143 ;Antilectura
                    00144 ;*********************************************
003B  1205          00145 _antilectura bcf 0x05,4 ;WR = 1
```

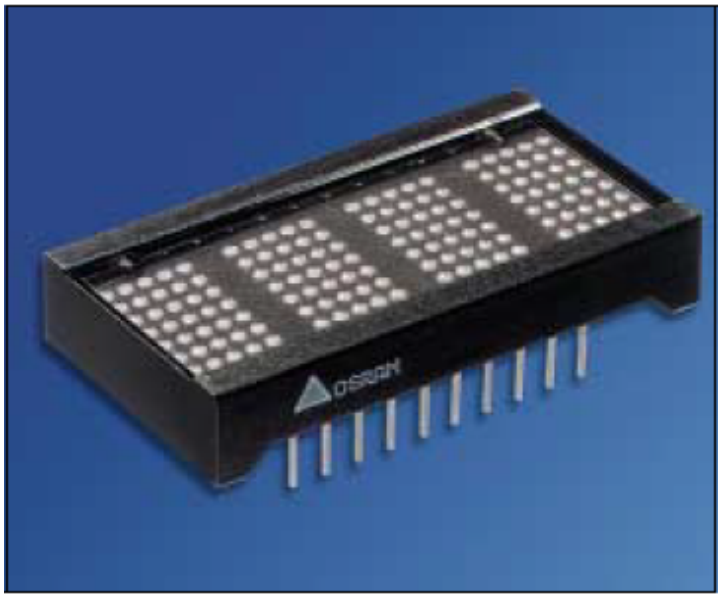
```
003C   1685          00146  bsf 0x05,5 ;RD = 0
003D   0008          00147  return
                     00148 ;*************************************************
                     00149 ;Lectura
                     00150 ;*************************************************
003E   1605          00151 _lectura bsf 0x05,4 ;WR = 0
003F   1285          00152  bcf 0x05,5 ;RD = 1
0040   0008          00153  return
                     00154 ;*************************************************
                     00155 ;Timer
                     00156 ;*************************************************
0041   3008   00157 _timer movlw 0x08 ;Aprox .5 seg
0042   00A0   00158  movwf 0x20

MPASM  5.51                 MATRIZ.ASM   4-30-2016  21:02:53        PAGE  4


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0043   0181          00159 _TLoop clrf 0x01 ;TMR0
0044   1D0B          00160  btfss 0x0B,2 ;TMR0IF=1????
0045   2844          00161  goto $-1
0046   110B          00162  bcf 0x0B,2 ;TMR0IF = 0
0047   0BA0          00163  decfsz 0x20,1
0048   2843          00164  goto _TLoop
0049   0008          00165  return
                     00166 ;*************************************************
                     00167 ;Mensaje
                     00168 ;*************************************************
0100                 00169  org 0x100
0100   0782   00170 _mensaje addwf 0x02,1
0101   3455 3441 3443 00171  dt "UACM"
     344D
                     00172  end

MPASM  5.51                 MATRIZ.ASM   4-30-2016  21:02:53        PAGE  5


SYMBOL TABLE
  LABEL                       VALUE

_TLoop                        00000043
__16F886                      00000001
_antilectura                  0000003B
_direccion                    00000031
_lectura                      0000003E
```

```
_m1                     00000028
_mensaje                  00000100
_timer                  00000041

MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXX------ ---------------- ---------------- ----------------
0100 : XXXXX---------- ---------------- ---------------- ----------------

All other memory blocks unused.

Program Memory Words Used:    79
Program Memory Words Free:  8113


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     5 suppressed
```

C1.3

Esquemático circuito pantalla led

- Características pantalla OSRAM PD4437:



- Módulo de 4 dígitos
- Funciones de control integradas
- Compatible con niveles TTL
- Carácter en formato ASCII
- Se puede conectar en cascada

Distribución de pines:

Pin	Function	Definition	Pin	Function	Definition
1	$\overline{RD}$	Active low, will enable a processor to read all registers in the display.	11	$\overline{WR}$	Write. Active low. If the device is selected, a low on the write input loads the data into memory.
2	CLK I/O	If CLK SEL (pin 3) is low, then expect an external clock source into this pin. If CLK SEL is high, then this pin will be the master or source into this pin. If CLK SEL is high, then this pin will be the master or source for all other devices which have CLK SEL low.	12	D7	Data Bus bit 7 (MSB).
3	$\overline{CLKSEL}$	CLOCK SELECT determines the action of pin 2. CLK I/O, see the section on Cascading for an example.	13	D6	Data Bus bit 6.
4	$\overline{RST}$	Reset. Used to synchronize blinking. Will not clear the display. The reset pulse should be less than 1 ms	14	D5	Data Bus bit 5.
5	CE1	Chip enable (active high).	15	D4	Data Bus bit 4.
6	$\overline{CE0}$	Chip enable (active low).	16	D3	Data Bus bit 3.
7	A2	Address input (MSB).	17	D2	Data Bus bit 2.
8	A1	Address input.	18	D1	Data Bus bit 1.
9	A0	Address input (LSB).	19	D0	Data Bus bit 0 (LSB).
10	GND	Ground.	20	V_{CC}	Positive power pin.

Características eléctricas:

Parameter	Limits				Conditions
	Min.	Typ.	Max.	Units	
V_{CC}	4.5	5.0	5.5	Volts	Nominal
I_{CC} (Blank)	—	2.5	3.5	mA	V_{CC}=5.0 V, A2=1, all other inputs low.
I_{CC} 80 LEDs/unit (100% Bright) PD243X PD353X PD443X	—	 115 145 150	 130 165 170	 mA mA mA	 V_{CC}=5.0 V V_{CC}=5.0 V V_{CC}=5.0 V
V_{IL}	—	—	0.8	Volts	V_{CC}=4.5 V to 5.5 V
V_{IH}	2.0	—	—	Volts	V_{CC}=4.5 V to 5.5 V
I_{IL} (except D0 to D7) (1)	25	—	100	µA	V_{CC}=4.5 V to 5.5 V, V_{IN}=0.8 V
V_{OL}	—	—	0.4	Volts	V_{CC}=4.5 V to 5.5 V
V_{OH}	2.4	—	—	Volts	V_{CC}=4.5 V to 5.5 V
I_{OH}	–8.9	—	—	mA	V_{CC}=4.5 V, V_{OH}=2.4 V
I_{OL}	1.6	—	—	mA	V_{CC}=4.5 V, V_{OL}=0.4 V
Data I/O Bus Loading	—	—	100	pF	—
Clock I/O Bus Loading	—	—	240	pF	—

1) D0 to D7 have no pull-up resistors so current is negligible.

- Modo de operación

Pines de control lectura/escritura:

El pin de lectura (pin 1) y escritura (pin 11) controlan el comportamiento del bus de datos, a continuación se muestra la tabla de operación.

RD!	WR!	Bus de datos (D7-D0)
X	X	Invalido
0	1	Salida
1	0	Entrada

Pines de activación:

El pin de activación CE0! (pin 6) habilita la escritura/lectura, mientras que el pin CE1 (pin 5) habilita el chip, a continuación se muestra su tabla de operación.

CE0!	CE1	RD!	WR!	Operación
0	1	0	0	Ninguna
1	X	X	X	Ninguna

X	0	X	X	Ninguna
X	X	1	1	Ninguna

Bus de direccionamiento:

Este bus está compuesto por los pines A2-A0 (pin 7-9), a continuación se muestra su tabla de operación.

A2	A1	A0	Contenido
0	X	X	Palabra de control
1	0	0	Dígito 0 (extremo Der.)
1	0	1	Dígito 1
1	1	0	Dígito 2
1	1	1	Dígito 3 (extremo Izq.)

Bus de datos:

Este bus esta compuesto por los pines D7-D0 (pin 12-19), entre las funciones de este bus se encuentra la transmisión/recepción de los datos pero también es el encargado de establecer la palabra de control de la pantalla.

Palabra de control:

Cuando el pin de dirección A2 (pin 7) se establece en estado bajo se accede a la palabra de control, esto aplica para los pines de dirección restantes, este registro se encarga de limpiar la pantalla, establecer el brillo, y de los atributos individuales de cada dígito.

Nivel de intensidad:

El nivel de intensidad es definido mediante los dos bits menos significativos del bus de datos (D0-D1), en la tabla siguiente se muestran los niveles de intensidad.

D7	D6	D5	D4	D3	D2	D1	D0	Brillo
0	0	X	X	X	X	0	0	0%
0	0	X	X	X	X	0	1	25%
0	0	X	X	X	X	1	0	50%
0	0	X	X	X	X	1	1	100%

Atributos:

Los bits D4,D3 y D2 controlan los efectos visuales de cada dígito que se muestra en pantalla, esta pantalla tiene cuatro diferentes atributos, para poder utilizarlos el bit D4 en la palabra de control (Atributos habilitados) debe estar siempre en alto de lo contrario los bits D3-D2 serán ignorados.

Los atributos no se destruyen, por ejemplo si el dígito tiene en alto el bit D7 el carácter es remplazado por el cursor, el carácter permanecerá en memoria y solo puede ser revertido de nuevo poniendo en bajo el bit D4, en la tabla siguiente se muestran todos los posibles atributos.

D7	D6	D5	D4	D3	D2	D1	D0	Atributo
X	X	X	0	X	X	B	B	Atributos deshabilitados
X	X	X	1	0	0	B	B	Muestra cursor en lugar de carácter
X	X	X	1	0	1	B	B	Parpadeo del carácter
X	X	X	1	1	0	B	B	Muestra cursor parpadeando en lugar de carácter
X	X	X	1	1	1	B	B	Alterna entre carácter y cursor

B -> Nivel de brillo seleccionado.

Parpadeo:

El bit D5 se encarga de causar un parpadeo general cuya frecuencia es de 2 Hz, en la siguiente tabla se muestran la tabla de control para este bit.

D7	D6	D5	D4	D3	D2	D1	D0	Acción
0	0	1	X	X	X	B	B	Parpadeo total de la pantalla

Test:

Si se pone en alto el bit D6 todos los leds que conforman la pantalla se encienden con un brillo del 50%, si el bit D6 se limpia todos los dígitos muestran los caracteres que tenía antes, en la siguiente tabla se muestran la tabla de control para este bit.

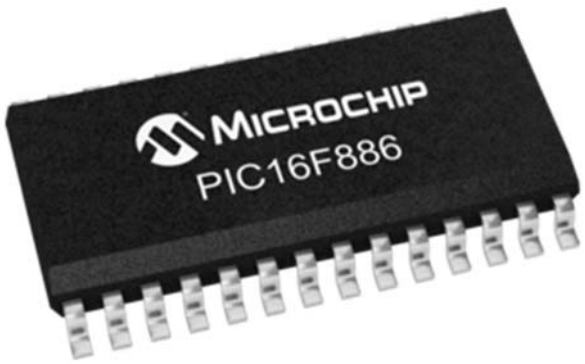
D7	D6	D5	D4	D3	D2	D1	D0	Acción
0	1	0	X	X	X	X	X	Test

Limpiar pantalla:

El bit D7 se encarga de eliminar los caracteres que se muestran en pantalla, y todos los datos que se encuentren guardados en memoria son puestos a cero, en la siguiente tabla se muestran la tabla de control para este bit.

D7	D6	D5	D4	D3	D2	D1	D0	Acción
1	0	X	X	X	X	X	X	La pantalla no muestra ningún carácter

- Características a destacar del microcontrolador PIC16F886

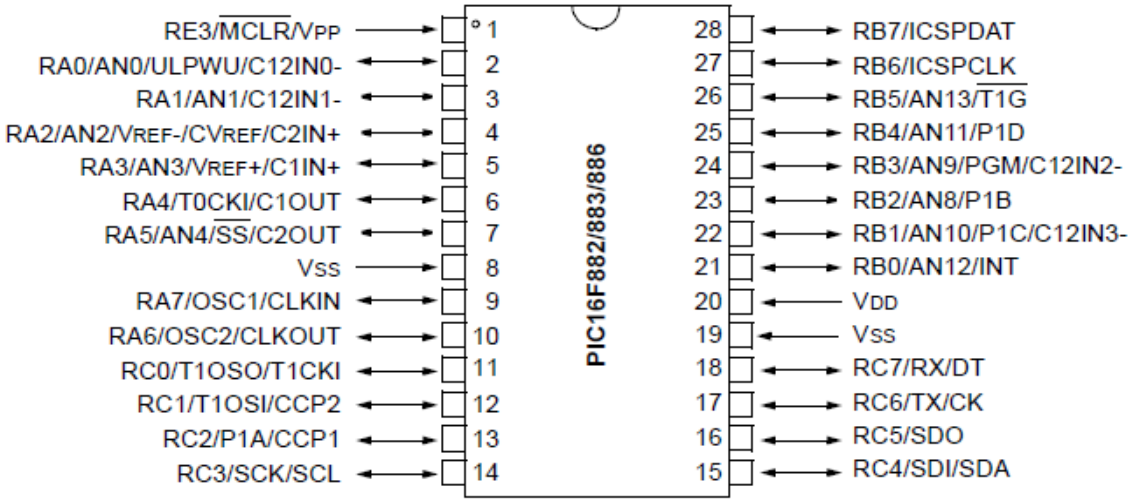


- Memoria programa de 14KB
- Memoria dato de 368 Bytes

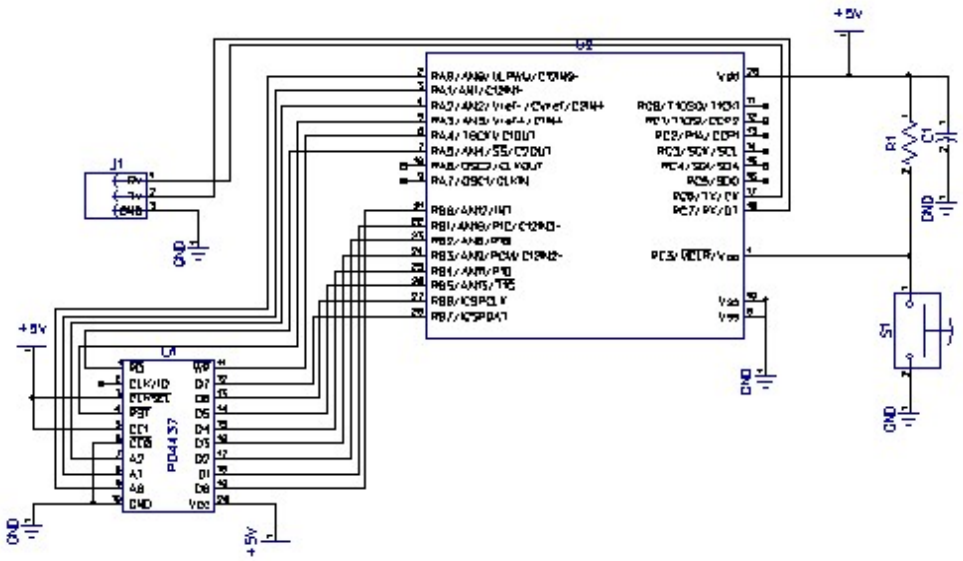
- Temporizadores 2 de 8 bits y 1 de 16 bits
- Oscilador interno hasta 8Mhz
- Módulo PWM con resolución de 8 bits

Distribución de pines:

28-pin PDIP, SOIC, SSOP



Esquemático del circuito utilizado en la pantalla led.



D1.1

Firmware DAQ

```
;  LST file generated by mikroListExporter - v.2.0
; Código DAQ
;E. David Rojas Serrano <edx@edx-twin3.org>
;http://www.edx-twin3.org
;-----------------------------------------------

;Address Opcode       ASM
0x0000          0x2EB3          GOTO      1715
_interrupt:
0x0004          0x087D          MOVF      R13, 0
0x0005          0x0020          MOVLB     0
0x0006          0x00A1          MOVWF     33
0x0007          0x018A          CLRF      PCLATH
0x0008          0x0183          CLRF      STATUS
;PuenteH.c,99 ::                void interrupt(void)
;PuenteH.c,102 ::               movlb 0x00
0x0009          0x0020          MOVLB     0
;PuenteH.c,103 ::               bcf 0x0B,2 //TMR0IF = 0;
0x000A          0x110B          BCF       INTCON, 2
;PuenteH.c,104 ::               decfsz _tiempo,1
0x000B          0x0BB2          DECFSZ    _tiempo, 1
;PuenteH.c,105 ::               retfie
0x000C          0x0009          RETFIE    0
;PuenteH.c,107 ::               Delay_us(288);
0x000D          0x30BF          MOVLW     191
0x000E          0x00FD          MOVWF     R13
L_interrupt0:
0x000F          0x0BFD          DECFSZ    R13, 1
0x0010          0x280F          GOTO      L_interrupt0
0x0011          0x0000          NOP
0x0012          0x0000          NOP
;PuenteH.c,108 ::               asm{retfie}
0x0013          0x0009          RETFIE    0
;PuenteH.c,109 ::               }
L_end_interrupt:
L__interrupt12:
0x0014          0x0821          MOVF      33, 0
0x0015          0x00FD          MOVWF     R13
0x0016          0x0009          RETFIE    %s
; end of _interrupt
_SETFOV32:
;__Lib_MathDouble.c,79 ::
;__Lib_MathDouble.c,86 ::
0x0017          0x14FB          BSF       R11, 1
```

```
;__Lib_MathDouble.c,87 ::
0x0018        0x1FFB        BTFSS     R11, 7
;__Lib_MathDouble.c,88 ::
0x0019        0x2821        GOTO      SETFOV32EEE
;__Lib_MathDouble.c,89 ::
0x001A        0x30FF        MOVLW     255
;__Lib_MathDouble.c,90 ::
0x001B        0x00F3        MOVWF     R3
;__Lib_MathDouble.c,91 ::
0x001C        0x00F2        MOVWF     R2
;__Lib_MathDouble.c,92 ::
0x001D        0x00F1        MOVWF     R1
;__Lib_MathDouble.c,93 ::
0x001E        0x00F0        MOVWF     R0
;__Lib_MathDouble.c,94 ::
0x001F        0x0DFA        RLF       R10, 1
;__Lib_MathDouble.c,95 ::
0x0020        0x0CF2        RRF       R2, 1
;__Lib_MathDouble.c,96 ::
SETFOV32EEE:
;__Lib_MathDouble.c,97 ::
0x0021        0x30FF        MOVLW     255
;__Lib_MathDouble.c,99 ::
L_end_SETFOV32:
0x0022        0x0008        RETURN
; end of _SETFOV32
_RES032:
;__Lib_MathDouble.c,123 ::
;__Lib_MathDouble.c,128 ::
0x0023        0x01F2        CLRF      R2
;__Lib_MathDouble.c,129 ::
0x0024        0x01F1        CLRF      R1
;__Lib_MathDouble.c,130 ::
0x0025        0x01F0        CLRF      R0
;__Lib_MathDouble.c,131 ::
0x0026        0x01F8        CLRF      R8
;__Lib_MathDouble.c,132 ::
0x0027        0x01F3        CLRF      R3
;__Lib_MathDouble.c,133 ::
0x0028        0x3000        MOVLW     0
;__Lib_MathDouble.c,135 ::
L_end_RES032:
0x0029        0x0008        RETURN
; end of _RES032
_SETFUN32:
;__Lib_MathDouble.c,138 ::
;__Lib_MathDouble.c,145 ::
```

```
0x002A          0x157B          BSF        R11, 2
;__Lib_MathDouble.c,146 ::
0x002B          0x1FFB          BTFSS      R11, 7
;__Lib_MathDouble.c,147 ::
0x002C          0x2834          GOTO       SETFUN32EEE
;__Lib_MathDouble.c,148 ::
0x002D          0x3001          MOVLW      1
;__Lib_MathDouble.c,149 ::
0x002E          0x00F3          MOVWF      R3
;__Lib_MathDouble.c,150 ::
0x002F          0x01F2          CLRF       R2
;__Lib_MathDouble.c,151 ::
0x0030          0x01F1          CLRF       R1
;__Lib_MathDouble.c,152 ::
0x0031          0x01F0          CLRF       R0
;__Lib_MathDouble.c,153 ::
0x0032          0x0DFA          RLF        R10, 1
;__Lib_MathDouble.c,154 ::
0x0033          0x0CF2          RRF        R2, 1
;__Lib_MathDouble.c,155 ::
SETFUN32EEE:
;__Lib_MathDouble.c,156 ::
0x0034          0x30FF          MOVLW      255
;__Lib_MathDouble.c,158 ::
L_end_SETFUN32:
0x0035          0x0008          RETURN
; end of _SETFUN32
_FIXSIGN32:
;__Lib_MathDouble.c,112 ::
;__Lib_MathDouble.c,116 ::
0x0036          0x1FFA          BTFSS      R10, 7
;__Lib_MathDouble.c,117 ::
0x0037          0x13F2          BCF        R2, 7
;__Lib_MathDouble.c,118 ::
0x0038          0x3000          MOVLW      0
;__Lib_MathDouble.c,120 ::
L_end_FIXSIGN32:
0x0039          0x0008          RETURN
; end of _FIXSIGN32
_Compare_Double:
;__Lib_MathDouble.c,2038 ::
;__Lib_MathDouble.c,2047 ::
;__Lib_MathDouble.c,2049 ::
0x003A          0x0020          MOVLB      0
;__Lib_MathDouble.c,2051 ::
0x003B          0x01FA          CLRF       R10
;__Lib_MathDouble.c,2052 ::
```

```
0x003C	0x0873	MOVF	R3, 0
;__Lib_MathDouble.c,2053 ::
0x003D	0x1D03	BTFSS	STATUS, 2
;__Lib_MathDouble.c,2054 ::
0x003E	0x147A	BSF	R10, 0
;__Lib_MathDouble.c,2055 ::
0x003F	0x0877	MOVF	R7, 0
;__Lib_MathDouble.c,2056 ::
0x0040	0x1D03	BTFSS	STATUS, 2
;__Lib_MathDouble.c,2057 ::
0x0041	0x14FA	BSF	R10, 1
;__Lib_MathDouble.c,2059 ::
0x0042	0x087A	MOVF	R10, 0
;__Lib_MathDouble.c,2060 ::
0x0043	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,2061 ::
0x0044	0x2885	GOTO	COMPARE_EQU
;__Lib_MathDouble.c,2062 ::
0x0045	0x3A03	XORLW	3
;__Lib_MathDouble.c,2063 ::
0x0046	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,2064 ::
0x0047	0x2850	GOTO	COMPARE_NOTZERO
;__Lib_MathDouble.c,2066 ::
0x0048	0x187A	BTFSC	R10, 0
;__Lib_MathDouble.c,2067 ::
0x0049	0x284D	GOTO	COMPARE_XNOTZERO
;__Lib_MathDouble.c,2069 ::
0x004A	0x1FF6	BTFSS	R6, 7
;__Lib_MathDouble.c,2070 ::
0x004B	0x2883	GOTO	COMPARE_LT
;__Lib_MathDouble.c,2071 ::
0x004C	0x2881	GOTO	COMPARE_GT
;__Lib_MathDouble.c,2072 ::
COMPARE_XNOTZERO:
;__Lib_MathDouble.c,2073 ::
0x004D	0x1FF2	BTFSS	R2, 7
;__Lib_MathDouble.c,2074 ::
0x004E	0x2881	GOTO	COMPARE_GT
;__Lib_MathDouble.c,2075 ::
0x004F	0x2883	GOTO	COMPARE_LT
;__Lib_MathDouble.c,2077 ::
COMPARE_NOTZERO:
;__Lib_MathDouble.c,2078 ::
0x0050	0x0872	MOVF	R2, 0
;__Lib_MathDouble.c,2079 ::
0x0051	0x3980	ANDLW	128
```

```
;__Lib_MathDouble.c,2080 ::
0x0052	0x00FA	MOVWF	R10
;__Lib_MathDouble.c,2081 ::
0x0053	0x13F2	BCF	R2, 7
;__Lib_MathDouble.c,2083 ::
0x0054	0x0876	MOVF	R6, 0
;__Lib_MathDouble.c,2084 ::
0x0055	0x3980	ANDLW	128
;__Lib_MathDouble.c,2085 ::
0x0056	0x13F6	BCF	R6, 7
;__Lib_MathDouble.c,2087 ::
0x0057	0x067A	XORWF	R10, 0
;__Lib_MathDouble.c,2088 ::
0x0058	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,2089 ::
0x0059	0x285E	GOTO	COMPARE_TESTALL
;__Lib_MathDouble.c,2091 ::
0x005A	0x087A	MOVF	R10, 0
;__Lib_MathDouble.c,2092 ::
0x005B	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,2093 ::
0x005C	0x2881	GOTO	COMPARE_GT
;__Lib_MathDouble.c,2094 ::
0x005D	0x2883	GOTO	COMPARE_LT
;__Lib_MathDouble.c,2096 ::
COMPARE_TESTALL:
;__Lib_MathDouble.c,2097 ::
0x005E	0x0877	MOVF	R7, 0
;__Lib_MathDouble.c,2098 ::
0x005F	0x0273	SUBWF	R3, 0
;__Lib_MathDouble.c,2099 ::
0x0060	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,2100 ::
0x0061	0x2865	GOTO	COMPARE_C1
;__Lib_MathDouble.c,2101 ::
0x0062	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,2102 ::
0x0063	0x2879	GOTO	GT_FALSE
;__Lib_MathDouble.c,2103 ::
0x0064	0x287D	GOTO	GT_TRUE
;__Lib_MathDouble.c,2104 ::
COMPARE_C1:
;__Lib_MathDouble.c,2105 ::
0x0065	0x0876	MOVF	R6, 0
;__Lib_MathDouble.c,2106 ::
0x0066	0x0272	SUBWF	R2, 0
;__Lib_MathDouble.c,2107 ::
```

```
0x0067	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,2108 ::
0x0068	0x286C	GOTO	COMPARE_C2
;__Lib_MathDouble.c,2109 ::
0x0069	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,2110 ::
0x006A	0x2879	GOTO	GT_FALSE
;__Lib_MathDouble.c,2111 ::
0x006B	0x287D	GOTO	GT_TRUE
;__Lib_MathDouble.c,2112 ::
COMPARE_C2:
;__Lib_MathDouble.c,2113 ::
0x006C	0x0875	MOVF	R5, 0
;__Lib_MathDouble.c,2114 ::
0x006D	0x0271	SUBWF	R1, 0
;__Lib_MathDouble.c,2115 ::
0x006E	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,2116 ::
0x006F	0x2873	GOTO	COMPARE_C3
;__Lib_MathDouble.c,2117 ::
0x0070	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,2118 ::
0x0071	0x2879	GOTO	GT_FALSE
;__Lib_MathDouble.c,2119 ::
0x0072	0x287D	GOTO	GT_TRUE
;__Lib_MathDouble.c,2120 ::
COMPARE_C3:
;__Lib_MathDouble.c,2121 ::
0x0073	0x0874	MOVF	R4, 0
;__Lib_MathDouble.c,2122 ::
0x0074	0x0270	SUBWF	R0, 0
;__Lib_MathDouble.c,2123 ::
0x0075	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,2124 ::
0x0076	0x2885	GOTO	COMPARE_EQU
;__Lib_MathDouble.c,2125 ::
0x0077	0x1803	BTFSC	STATUS, 0
;__Lib_MathDouble.c,2126 ::
0x0078	0x287D	GOTO	GT_TRUE
;__Lib_MathDouble.c,2128 ::
GT_FALSE:
;__Lib_MathDouble.c,2129 ::
0x0079	0x08FA	MOVF	R10, 1
;__Lib_MathDouble.c,2130 ::
0x007A	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,2131 ::
0x007B	0x2883	GOTO	COMPARE_LT
```

```
;__Lib_MathDouble.c,2132 ::
0x007C        0x2881        GOTO      COMPARE_GT
;__Lib_MathDouble.c,2133 ::
GT_TRUE:
;__Lib_MathDouble.c,2134 ::
0x007D        0x08FA        MOVF      R10, 1
;__Lib_MathDouble.c,2135 ::
0x007E        0x1903        BTFSC     STATUS, 2
;__Lib_MathDouble.c,2136 ::
0x007F        0x2881        GOTO      COMPARE_GT
;__Lib_MathDouble.c,2137 ::
0x0080        0x2883        GOTO      COMPARE_LT
;__Lib_MathDouble.c,2139 ::
COMPARE_GT:
;__Lib_MathDouble.c,2142 ::
0x0081        0x1403        BSF       STATUS, 0
;__Lib_MathDouble.c,2143 ::
0x0082        0x0008        RETURN
;__Lib_MathDouble.c,2144 ::
COMPARE_LT:
;__Lib_MathDouble.c,2147 ::
0x0083        0x1003        BCF       STATUS, 0
;__Lib_MathDouble.c,2148 ::
0x0084        0x0008        RETURN
;__Lib_MathDouble.c,2149 ::
COMPARE_EQU:
;__Lib_MathDouble.c,2152 ::
0x0085        0x1403        BSF       STATUS, 0
;__Lib_MathDouble.c,2153 ::
0x0086        0x1503        BSF       STATUS, 2
;__Lib_MathDouble.c,2154 ::
0x0087        0x0008        RETURN
;__Lib_MathDouble.c,2156 ::
L_end_Compare_Double:
0x0088        0x0008        RETURN
; end of _Compare_Double
_strcpy:
;__Lib_CString.c,126 ::
;__Lib_CString.c,129 ::
0x0089        0x0020        MOVLB     0
0x008A        0x085A        MOVF      FARG_strcpy_to, 0
0x008B        0x00F4        MOVWF     R4
0x008C        0x085B        MOVF      FARG_strcpy_to+1, 0
0x008D        0x00F5        MOVWF     R4+1
;__Lib_CString.c,130 ::
L_strcpy34:
0x008E        0x0874        MOVF      R4, 0
```

```
0x008F      0x00F2      MOVWF    R2
0x0090      0x0875      MOVF     R4+1, 0
0x0091      0x00F3      MOVWF    R3
0x0092      0x0AF4      INCF     R4, 1
0x0093      0x1903      BTFSC    STATUS, 2
0x0094      0x0AF5      INCF     R4+1, 1
0x0095      0x085C      MOVF     FARG_strcpy_from, 0
0x0096      0x00F0      MOVWF    R0
0x0097      0x085D      MOVF     FARG_strcpy_from+1, 0
0x0098      0x00F1      MOVWF    R1
0x0099      0x0ADC      INCF     FARG_strcpy_from, 1
0x009A      0x1903      BTFSC    STATUS, 2
0x009B      0x0ADD      INCF     FARG_strcpy_from+1, 1
0x009C      0x0870      MOVF     R0, 0
0x009D      0x0084      MOVWF    FSR0
0x009E      0x0871      MOVF     R1, 0
0x009F      0x0085      MOVWF    FSR0H
0x00A0      0x0872      MOVF     R2, 0
0x00A1      0x0086      MOVWF    FSR1
0x00A2      0x0873      MOVF     R3, 0
0x00A3      0x0087      MOVWF    FSR1H
0x00A4      0x0800      MOVF     INDF0, 0
0x00A5      0x0081      MOVWF    INDF1
0x00A6      0x0872      MOVF     R2, 0
0x00A7      0x0084      MOVWF    FSR0
0x00A8      0x0873      MOVF     R3, 0
0x00A9      0x0085      MOVWF    FSR0H
0x00AA      0x0800      MOVF     INDF0, 0
0x00AB      0x1903      BTFSC    STATUS, 2
0x00AC      0x28AE      GOTO     L_strcpy35
;__Lib_CString.c,131 ::
0x00AD      0x288E      GOTO     L_strcpy34
L_strcpy35:
;__Lib_CString.c,132 ::
0x00AE      0x085A      MOVF     FARG_strcpy_to, 0
0x00AF      0x00F0      MOVWF    R0
0x00B0      0x085B      MOVF     FARG_strcpy_to+1, 0
0x00B1      0x00F1      MOVWF    R1
;__Lib_CString.c,133 ::
L_end_strcpy:
0x00B2      0x0008      RETURN
; end of _strcpy
_UART1_Write:
;__Lib_UART_b21.c,53 ::
;__Lib_UART_b21.c,54 ::
L_UART1_Write3:
0x00B3      0x0023      MOVLB    3
```

```
0x00B4  0x189E  BTFSC   TXSTA, 1
0x00B5  0x28B8  GOTO    L_UART1_Write4
;__Lib_UART_b21.c,55 ::
0x00B6  0x0000  NOP
0x00B7  0x28B3  GOTO    L_UART1_Write3
L_UART1_Write4:
;__Lib_UART_b21.c,56 ::
0x00B8  0x0020  MOVLB   0
0x00B9  0x085A  MOVF    FARG_UART1_Write_data_, 0
0x00BA  0x0023  MOVLB   3
0x00BB  0x009A  MOVWF   TXREG
;__Lib_UART_b21.c,57 ::
L_end_UART1_Write:
0x00BC  0x0008  RETURN
; end of _UART1_Write
_SETFDZ32:
;__Lib_MathDouble.c,102 ::
;__Lib_MathDouble.c,105 ::
0x00BD  0x15FB  BSF     R11, 3
;__Lib_MathDouble.c,106 ::
0x00BE  0x30FF  MOVLW   255
;__Lib_MathDouble.c,108 ::
L_end_SETFDZ32:
0x00BF  0x0008  RETURN
; end of _SETFDZ32
_Mul_32x32_FP:
;__Lib_MathDouble.c,608 ::
;__Lib_MathDouble.c,620 ::
;__Lib_MathDouble.c,622 ::
0x00C0  0x0020  MOVLB   0
;__Lib_MathDouble.c,624 ::
0x00C1  0x01FB  CLRF    R11
;__Lib_MathDouble.c,625 ::
0x00C2  0x0873  MOVF    R3, 0
;__Lib_MathDouble.c,626 ::
0x00C3  0x1D03  BTFSS   STATUS, 2
;__Lib_MathDouble.c,627 ::
0x00C4  0x0877  MOVF    R7, 0
;__Lib_MathDouble.c,628 ::
0x00C5  0x1903  BTFSC   STATUS, 2
;__Lib_MathDouble.c,629 ::
0x00C6  0x2916  GOTO    JPMRES032
;__Lib_MathDouble.c,631 ::
0x00C7  0x0872  MOVF    R2, 0
;__Lib_MathDouble.c,632 ::
0x00C8  0x0676  XORWF   R6, 0
;__Lib_MathDouble.c,633 ::
```

```
0x00C9	0x00FA	MOVWF	R10
;__Lib_MathDouble.c,634 ::
0x00CA	0x0877	MOVF	R7, 0
;__Lib_MathDouble.c,635 ::
0x00CB	0x07F3	ADDWF	R3, 1
;__Lib_MathDouble.c,636 ::
0x00CC	0x307E	MOVLW	126
;__Lib_MathDouble.c,637 ::
0x00CD	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,638 ::
0x00CE	0x28D3	GOTO	MTUN32
;__Lib_MathDouble.c,639 ::
0x00CF	0x02F3	SUBWF	R3, 1
;__Lib_MathDouble.c,640 ::
0x00D0	0x1803	BTFSC	STATUS, 0
;__Lib_MathDouble.c,641 ::
0x00D1	0x2919	GOTO	JPMSETFOV32
;__Lib_MathDouble.c,642 ::
0x00D2	0x28D6	GOTO	MOK32
;__Lib_MathDouble.c,643 ::
MTUN32:
;__Lib_MathDouble.c,644 ::
0x00D3	0x02F3	SUBWF	R3, 1
;__Lib_MathDouble.c,645 ::
0x00D4	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,646 ::
0x00D5	0x291C	GOTO	JPMSETFUN32
;__Lib_MathDouble.c,647 ::
MOK32:
;__Lib_MathDouble.c,648 ::
0x00D6	0x0872	MOVF	R2, 0
;__Lib_MathDouble.c,649 ::
0x00D7	0x00F8	MOVWF	R8
;__Lib_MathDouble.c,650 ::
0x00D8	0x0871	MOVF	R1, 0
;__Lib_MathDouble.c,651 ::
0x00D9	0x00FC	MOVWF	R12
;__Lib_MathDouble.c,652 ::
0x00DA	0x0870	MOVF	R0, 0
;__Lib_MathDouble.c,653 ::
0x00DB	0x00FD	MOVWF	R13
;__Lib_MathDouble.c,654 ::
0x00DC	0x17F8	BSF	R8, 7
;__Lib_MathDouble.c,655 ::
0x00DD	0x17F6	BSF	R6, 7
;__Lib_MathDouble.c,656 ::
0x00DE	0x1003	BCF	STATUS, 0
```

```
;__Lib_MathDouble.c,657 ::
0x00DF        0x01F2        CLRF      R2
;__Lib_MathDouble.c,658 ::
0x00E0        0x01F1        CLRF      R1
;__Lib_MathDouble.c,659 ::
0x00E1        0x01F0        CLRF      R0
;__Lib_MathDouble.c,660 ::
0x00E2        0x3018        MOVLW     24
;__Lib_MathDouble.c,661 ::
0x00E3        0x00A0        MOVWF     __math_tempbD
;__Lib_MathDouble.c,662 ::
MLOOP32:
;__Lib_MathDouble.c,663 ::
0x00E4        0x1C7D        BTFSS     R13, 0
;__Lib_MathDouble.c,664 ::
0x00E5        0x28F0        GOTO      MNOADD32
;__Lib_MathDouble.c,666 ::
0x00E6        0x0874        MOVF      R4, 0
;__Lib_MathDouble.c,667 ::
0x00E7        0x07F0        ADDWF     R0, 1
;__Lib_MathDouble.c,668 ::
0x00E8        0x0875        MOVF      R5, 0
;__Lib_MathDouble.c,669 ::
0x00E9        0x1803        BTFSC     STATUS, 0
;__Lib_MathDouble.c,670 ::
0x00EA        0x0F75        INCFSZ    R5, 0
;__Lib_MathDouble.c,671 ::
0x00EB        0x07F1        ADDWF     R1, 1
;__Lib_MathDouble.c,672 ::
0x00EC        0x0876        MOVF      R6, 0
;__Lib_MathDouble.c,673 ::
0x00ED        0x1803        BTFSC     STATUS, 0
;__Lib_MathDouble.c,674 ::
0x00EE        0x0F76        INCFSZ    R6, 0
;__Lib_MathDouble.c,675 ::
0x00EF        0x07F2        ADDWF     R2, 1
;__Lib_MathDouble.c,676 ::
MNOADD32:
;__Lib_MathDouble.c,677 ::
0x00F0        0x0CF2        RRF       R2, 1
;__Lib_MathDouble.c,678 ::
0x00F1        0x0CF1        RRF       R1, 1
;__Lib_MathDouble.c,679 ::
0x00F2        0x0CF0        RRF       R0, 1
;__Lib_MathDouble.c,680 ::
0x00F3        0x0CF8        RRF       R8, 1
;__Lib_MathDouble.c,681 ::
```

```
0x00F4          0x0CFC          RRF         R12, 1
;__Lib_MathDouble.c,682 ::
0x00F5          0x0CFD          RRF         R13, 1
;__Lib_MathDouble.c,683 ::
0x00F6          0x1003          BCF         STATUS, 0
;__Lib_MathDouble.c,684 ::
0x00F7          0x0BA0          DECFSZ      __math_tempbD, 1
;__Lib_MathDouble.c,685 ::
0x00F8          0x28E4          GOTO        MLOOP32
;__Lib_MathDouble.c,686 ::
0x00F9          0x1BF2          BTFSC       R2, 7
;__Lib_MathDouble.c,687 ::
0x00FA          0x2900          GOTO        MROUND32
;__Lib_MathDouble.c,688 ::
0x00FB          0x0DF8          RLF         R8, 1
;__Lib_MathDouble.c,689 ::
0x00FC          0x0DF0          RLF         R0, 1
;__Lib_MathDouble.c,690 ::
0x00FD          0x0DF1          RLF         R1, 1
;__Lib_MathDouble.c,691 ::
0x00FE          0x0DF2          RLF         R2, 1
;__Lib_MathDouble.c,692 ::
0x00FF          0x03F3          DECF        R3, 1
;__Lib_MathDouble.c,693 ::
MROUND32:
;__Lib_MathDouble.c,694 ::
0x0100          0x1B7B          BTFSC       R11, 6
;__Lib_MathDouble.c,695 ::
0x0101          0x1C70          BTFSS       R0, 0
;__Lib_MathDouble.c,696 ::
0x0102          0x2912          GOTO        MUL32OK
;__Lib_MathDouble.c,697 ::
0x0103          0x1FF8          BTFSS       R8, 7
;__Lib_MathDouble.c,698 ::
0x0104          0x2912          GOTO        MUL32OK
;__Lib_MathDouble.c,699 ::
0x0105          0x0AF0          INCF        R0, 1
;__Lib_MathDouble.c,700 ::
0x0106          0x1903          BTFSC       STATUS, 2
;__Lib_MathDouble.c,701 ::
0x0107          0x0AF1          INCF        R1, 1
;__Lib_MathDouble.c,702 ::
0x0108          0x1903          BTFSC       STATUS, 2
;__Lib_MathDouble.c,703 ::
0x0109          0x0AF2          INCF        R2, 1
;__Lib_MathDouble.c,704 ::
0x010A          0x1D03          BTFSS       STATUS, 2
```

```
;__Lib_MathDouble.c,705 ::
0x010B        0x2912        GOTO      MUL32OK
;__Lib_MathDouble.c,706 ::
0x010C        0x0CF2        RRF       R2, 1
;__Lib_MathDouble.c,707 ::
0x010D        0x0CF1        RRF       R1, 1
;__Lib_MathDouble.c,708 ::
0x010E        0x0CF0        RRF       R0, 1
;__Lib_MathDouble.c,709 ::
0x010F        0x0AF3        INCF      R3, 1
;__Lib_MathDouble.c,710 ::
0x0110        0x1903        BTFSC     STATUS, 2
;__Lib_MathDouble.c,711 ::
0x01110x2919          GOTO      JPMSETFOV32
;__Lib_MathDouble.c,712 ::
MUL32OK:
;__Lib_MathDouble.c,713 ::
0x0112        0x1FFA        BTFSS     R10, 7
;__Lib_MathDouble.c,714 ::
0x0113        0x13F2        BCF       R2, 7
;__Lib_MathDouble.c,715 ::
0x0114        0x3000        MOVLW     0
;__Lib_MathDouble.c,717 ::
0x0115        0x291E        GOTO      MUL32EEE
;__Lib_MathDouble.c,718 ::
JPMRES032:
;__Lib_MathDouble.c,719 ::
0x0116        0x3001        MOVLW     1
;__Lib_MathDouble.c,722 ::
0x0117        0x2023        CALL      _RES032
;__Lib_MathDouble.c,724 ::
0x0118        0x291E        GOTO      MUL32EEE
;__Lib_MathDouble.c,725 ::
JPMSETFOV32:
;__Lib_MathDouble.c,726 ::
0x0119        0x3002        MOVLW     2
;__Lib_MathDouble.c,729 ::
0x011A        0x2017        CALL      _SETFOV32
;__Lib_MathDouble.c,731 ::
0x011B        0x291E        GOTO      MUL32EEE
;__Lib_MathDouble.c,732 ::
JPMSETFUN32:
;__Lib_MathDouble.c,733 ::
0x011C        0x3004        MOVLW     4
;__Lib_MathDouble.c,736 ::
0x011D        0x202A        CALL      _SETFUN32
;__Lib_MathDouble.c,738 ::
```

```
MUL32EEE:
;__Lib_MathDouble.c,739 ::
0x011E          0x0000          NOP
;__Lib_MathDouble.c,746 ::
L_end_Mul_32x32_FP:
0x011F          0x0008          RETURN
; end of _Mul_32x32_FP
_NRM4032:
;__Lib_MathDouble.c,245 ::
;__Lib_MathDouble.c,252 ::
;__Lib_MathDouble.c,254 ::
0x0120          0x187C          BTFSC       R12, 0
;__Lib_MathDouble.c,255 ::
0x0121          0x2956          GOTO        NRMRND4032
;__Lib_MathDouble.c,256 ::
0x0122          0x18FC          BTFSC       R12, 1
;__Lib_MathDouble.c,257 ::
0x0123          0x2927          GOTO        NRM4032
;__Lib_MathDouble.c,258 ::
0x0124          0x197C          BTFSC       R12, 2
;__Lib_MathDouble.c,259 ::
0x0125          0x2975          GOTO        JMPSETFOV32
;__Lib_MathDouble.c,260 ::
0x0126          0x2920          GOTO        $-6
;__Lib_MathDouble.c,261 ::
NRM4032:
;__Lib_MathDouble.c,262 ::
0x0127          0x01A0          CLRF        __math_tempbD
;__Lib_MathDouble.c,263 ::
0x0128          0x0872          MOVF        R2, 0
;__Lib_MathDouble.c,264 ::
0x0129          0x1D03          BTFSS       STATUS, 2
;__Lib_MathDouble.c,265 ::
0x012A          0x2947          GOTO        NORM4032
;__Lib_MathDouble.c,266 ::
0x012B          0x0871          MOVF        R1, 0
;__Lib_MathDouble.c,267 ::
0x012C          0x00F2          MOVWF       R2
;__Lib_MathDouble.c,268 ::
0x012D          0x0870          MOVF        R0, 0
;__Lib_MathDouble.c,269 ::
0x012E          0x00F1          MOVWF       R1
;__Lib_MathDouble.c,270 ::
0x012F          0x0878          MOVF        R8, 0
;__Lib_MathDouble.c,271 ::
0x0130          0x00F0          MOVWF       R0
;__Lib_MathDouble.c,272 ::
```

```
0x0131	0x01F8	CLRF	R8
;__Lib_MathDouble.c,273 ::
0x0132	0x15A0	BSF	__math_tempbD, 3
;__Lib_MathDouble.c,275 ::
0x0133	0x0872	MOVF	R2, 0
;__Lib_MathDouble.c,276 ::
0x0134	0x1D03	BTFSS	STATUS, 2
;__Lib_MathDouble.c,277 ::
0x0135	0x2947	GOTO	NORM4032
;__Lib_MathDouble.c,278 ::
0x0136	0x0871	MOVF	R1, 0
;__Lib_MathDouble.c,279 ::
0x0137	0x00F2	MOVWF	R2
;__Lib_MathDouble.c,280 ::
0x0138	0x0870	MOVF	R0, 0
;__Lib_MathDouble.c,281 ::
0x0139	0x00F1	MOVWF	R1
;__Lib_MathDouble.c,282 ::
0x013A	0x01F0	CLRF	R0
;__Lib_MathDouble.c,283 ::
0x013B	0x11A0	BCF	__math_tempbD, 3
;__Lib_MathDouble.c,284 ::
0x013C	0x1620	BSF	__math_tempbD, 4
;__Lib_MathDouble.c,286 ::
0x013D	0x0872	MOVF	R2, 0
;__Lib_MathDouble.c,287 ::
0x013E	0x1D03	BTFSS	STATUS, 2
;__Lib_MathDouble.c,288 ::
0x013F	0x2947	GOTO	NORM4032
;__Lib_MathDouble.c,289 ::
0x0140	0x0871	MOVF	R1, 0
;__Lib_MathDouble.c,290 ::
0x0141	0x00F2	MOVWF	R2
;__Lib_MathDouble.c,291 ::
0x0142	0x01F1	CLRF	R1
;__Lib_MathDouble.c,292 ::
0x0143	0x15A0	BSF	__math_tempbD, 3
;__Lib_MathDouble.c,294 ::
0x0144	0x0872	MOVF	R2, 0
;__Lib_MathDouble.c,295 ::
0x0145	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,296 ::
0x0146	0x2969	GOTO	JMPRES032
;__Lib_MathDouble.c,297 ::
NORM4032:
;__Lib_MathDouble.c,298 ::
0x0147	0x0820	MOVF	__math_tempbD, 0
```

```
;__Lib_MathDouble.c,299 ::
0x0148        0x02F3        SUBWF     R3, 1
;__Lib_MathDouble.c,300 ::
0x0149        0x1D03        BTFSS     STATUS, 2
;__Lib_MathDouble.c,301 ::
0x014A        0x1C03        BTFSS     STATUS, 0
;__Lib_MathDouble.c,302 ::
0x014B        0x296D        GOTO      JMPSETFUN32
;__Lib_MathDouble.c,303 ::
0x014C        0x1003        BCF       STATUS, 0
;__Lib_MathDouble.c,304 ::
NORM4032A:
;__Lib_MathDouble.c,305 ::
0x014D        0x1BF2        BTFSC     R2, 7
;__Lib_MathDouble.c,306 ::
0x014E        0x2956        GOTO      NRMRND4032
;__Lib_MathDouble.c,307 ::
0x014F        0x0DF8        RLF       R8, 1
;__Lib_MathDouble.c,308 ::
0x0150        0x0DF0        RLF       R0, 1
;__Lib_MathDouble.c,309 ::
0x0151        0x0DF1        RLF       R1, 1
;__Lib_MathDouble.c,310 ::
0x0152        0x0DF2        RLF       R2, 1
;__Lib_MathDouble.c,311 ::
0x0153        0x0BF3        DECFSZ    R3, 1
;__Lib_MathDouble.c,312 ::
0x0154        0x294D        GOTO      NORM4032A
;__Lib_MathDouble.c,313 ::
0x0155        0x296D        GOTO      JMPSETFUN32
;__Lib_MathDouble.c,314 ::
NRMRND4032:
;__Lib_MathDouble.c,315 ::
0x0156        0x1B7B        BTFSC     R11, 6
;__Lib_MathDouble.c,316 ::
0x0157        0x1C70        BTFSS     R0, 0
;__Lib_MathDouble.c,317 ::
0x0158        0x2971        GOTO      JMPFIXSIGN32
;__Lib_MathDouble.c,318 ::
0x0159        0x1FF8        BTFSS     R8, 7
;__Lib_MathDouble.c,319 ::
0x015A        0x2971        GOTO      JMPFIXSIGN32
;__Lib_MathDouble.c,320 ::
0x015B        0x0AF0        INCF      R0, 1
;__Lib_MathDouble.c,321 ::
0x015C        0x1903        BTFSC     STATUS, 2
;__Lib_MathDouble.c,322 ::
```

```
0x015D	0x0AF1	INCF	R1, 1
;__Lib_MathDouble.c,323 ::
0x015E	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,324 ::
0x015F	0x0AF2	INCF	R2, 1
;__Lib_MathDouble.c,325 ::
0x0160	0x1D03	BTFSS	STATUS, 2
;__Lib_MathDouble.c,326 ::
0x0161	0x2971	GOTO	JMPFIXSIGN32
;__Lib_MathDouble.c,327 ::
0x0162	0x0CF2	RRF	R2, 1
;__Lib_MathDouble.c,328 ::
0x0163	0x0CF1	RRF	R1, 1
;__Lib_MathDouble.c,329 ::
0x0164	0x0CF0	RRF	R0, 1
;__Lib_MathDouble.c,330 ::
0x0165	0x0AF3	INCF	R3, 1
;__Lib_MathDouble.c,331 ::
0x0166	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,332 ::
0x0167	0x2975	GOTO	JMPSETFOV32
;__Lib_MathDouble.c,333 ::
0x0168	0x2971	GOTO	JMPFIXSIGN32
;__Lib_MathDouble.c,334 ::
JMPRES032:
;__Lib_MathDouble.c,335 ::
0x0169	0x3001	MOVLW	1
;__Lib_MathDouble.c,336 ::
0x016A	0x00FC	MOVWF	R12
;__Lib_MathDouble.c,338 ::
0x016B	0x2023	CALL	_RES032
;__Lib_MathDouble.c,340 ::
0x016C	0x2978	GOTO	NRM4032EEE
;__Lib_MathDouble.c,341 ::
JMPSETFUN32:
;__Lib_MathDouble.c,342 ::
0x016D	0x3002	MOVLW	2
;__Lib_MathDouble.c,343 ::
0x016E	0x00FC	MOVWF	R12
;__Lib_MathDouble.c,345 ::
0x016F	0x202A	CALL	_SETFUN32
;__Lib_MathDouble.c,347 ::
0x0170	0x2978	GOTO	NRM4032EEE
;__Lib_MathDouble.c,348 ::
JMPFIXSIGN32:
;__Lib_MathDouble.c,349 ::
0x0171	0x3004	MOVLW	4
```

```
;__Lib_MathDouble.c,350 ::
0x0172	0x00FC	MOVWF	R12
;__Lib_MathDouble.c,352 ::
0x0173	0x2036	CALL	_FIXSIGN32
;__Lib_MathDouble.c,354 ::
0x0174	0x2978	GOTO	NRM4032EEE
;__Lib_MathDouble.c,355 ::
JMPSETFOV32:
;__Lib_MathDouble.c,356 ::
0x0175	0x3008	MOVLW	8
;__Lib_MathDouble.c,357 ::
0x0176	0x00FC	MOVWF	R12
;__Lib_MathDouble.c,359 ::
0x0177	0x2017	CALL	_SETFOV32
;__Lib_MathDouble.c,361 ::
NRM4032EEE:
;__Lib_MathDouble.c,362 ::
0x0178	0x0000	NOP
;__Lib_MathDouble.c,370 ::
L_end_NRM4032:
0x0179	0x0008	RETURN
; end of _NRM4032
_Mul_32x32_U:
;__Lib_Math.c,1765 ::
;__Lib_Math.c,1780 ::
0x017A	0x0020	MOVLB	0
;__Lib_Math.c,1781 ::
0x017B	0x3022	MOVLW	34
;__Lib_Math.c,1782 ::
0x017C	0x00FC	MOVWF	R12
;__Lib_Math.c,1783 ::
0x017D	0x01F8	CLRF	R8
;__Lib_Math.c,1784 ::
0x017E	0x01F9	CLRF	R9
;__Lib_Math.c,1785 ::
0x017F	0x01FA	CLRF	R10
;__Lib_Math.c,1786 ::
0x0180	0x01FB	CLRF	R11
;__Lib_Math.c,1788 ::
_NEXT:
;__Lib_Math.c,1792 ::
0x0181	0x03FC	DECF	R12, 1
;__Lib_Math.c,1793 ::
0x0182	0x1903	BTFSC	STATUS, 2
;__Lib_Math.c,1794 ::
0x0183	0x29AF	GOTO	_EXIT2
;__Lib_Math.c,1795 ::
```

```
0x0184          0x1003          BCF       STATUS, 0
;__Lib_Math.c,1797 ::
_LOOP:
;__Lib_Math.c,1806 ::
0x0185          0x0CFB          RRF       R11, 1
;__Lib_Math.c,1807 ::
0x0186          0x0CFA          RRF       R10, 1
;__Lib_Math.c,1808 ::
0x0187          0x0CF9          RRF       R9, 1
;__Lib_Math.c,1809 ::
0x0188          0x0CF8          RRF       R8, 1
;__Lib_Math.c,1810 ::
0x0189          0x0CF3          RRF       R3, 1
;__Lib_Math.c,1811 ::
0x018A          0x0CF2          RRF       R2, 1
;__Lib_Math.c,1812 ::
0x018B          0x0CF1          RRF       R1, 1
;__Lib_Math.c,1813 ::
0x018C          0x0CF0          RRF       R0, 1
;__Lib_Math.c,1818 ::
0x018D          0x1C03          BTFSS     STATUS, 0
;__Lib_Math.c,1819 ::
0x018E          0x2981          GOTO      _NEXT
;__Lib_Math.c,1820 ::
0x018F          0x03FC          DECF      R12, 1
;__Lib_Math.c,1821 ::
0x0190          0x1903          BTFSC     STATUS, 2
;__Lib_Math.c,1822 ::
0x0191          0x29A1          GOTO      _EXIT1
;__Lib_Math.c,1829 ::
0x0192          0x0874          MOVF      R4, 0
;__Lib_Math.c,1830 ::
0x0193          0x07F8          ADDWF     R8, 1
;__Lib_Math.c,1831 ::
0x0194          0x0875          MOVF      R5, 0
;__Lib_Math.c,1832 ::
0x0195          0x1803          BTFSC     STATUS, 0
;__Lib_Math.c,1833 ::
0x0196          0x0F75          INCFSZ    R5, 0
;__Lib_Math.c,1834 ::
0x0197          0x07F9          ADDWF     R9, 1
;__Lib_Math.c,1835 ::
0x0198          0x0876          MOVF      R6, 0
;__Lib_Math.c,1836 ::
0x0199          0x1803          BTFSC     STATUS, 0
;__Lib_Math.c,1837 ::
0x019A          0x0F76          INCFSZ    R6, 0
```

```
;__Lib_Math.c,1838 ::
0x019B      0x07FA      ADDWF    R10, 1
;__Lib_Math.c,1839 ::
0x019C      0x0877      MOVF     R7, 0
;__Lib_Math.c,1840 ::
0x019D      0x1803      BTFSC    STATUS, 0
;__Lib_Math.c,1841 ::
0x019E      0x0F77      INCFSZ   R7, 0
;__Lib_Math.c,1842 ::
0x019F      0x07FB      ADDWF    R11, 1
;__Lib_Math.c,1844 ::
0x01A0      0x2985      GOTO     _LOOP
;__Lib_Math.c,1846 ::
_EXIT1:
;__Lib_Math.c,1851 ::
0x01A1      0x0874      MOVF     R4, 0
;__Lib_Math.c,1852 ::
0x01A2      0x07F8      ADDWF    R8, 1
;__Lib_Math.c,1853 ::
0x01A3      0x0875      MOVF     R5, 0
;__Lib_Math.c,1854 ::
0x01A4      0x1803      BTFSC    STATUS, 0
;__Lib_Math.c,1855 ::
0x01A5      0x0F75      INCFSZ   R5, 0
;__Lib_Math.c,1856 ::
0x01A6      0x07F9      ADDWF    R9, 1
;__Lib_Math.c,1857 ::
0x01A7      0x0876      MOVF     R6, 0
;__Lib_Math.c,1858 ::
0x01A8      0x1803      BTFSC    STATUS, 0
;__Lib_Math.c,1859 ::
0x01A9      0x0F76      INCFSZ   R6, 0
;__Lib_Math.c,1860 ::
0x01AA      0x07FA      ADDWF    R10, 1
;__Lib_Math.c,1861 ::
0x01AB      0x0877      MOVF     R7, 0
;__Lib_Math.c,1862 ::
0x01AC      0x1803      BTFSC    STATUS, 0
;__Lib_Math.c,1863 ::
0x01AD      0x0F77      INCFSZ   R7, 0
;__Lib_Math.c,1864 ::
0x01AE      0x07FB      ADDWF    R11, 1
;__Lib_Math.c,1867 ::
_EXIT2:
;__Lib_Math.c,1869 ::
L_end_Mul_32x32_U:
0x01AF      0x0008      RETURN
```

```
; end of _Mul_32x32_U
_srand:
;__Lib_CStdlib.c,301 ::
;__Lib_CStdlib.c,302 ::
0x01B0          0x0020          MOVLB     0
0x01B1          0x085A          MOVF      FARG_srand_x, 0
0x01B2          0x00B5          MOVWF     __Lib_CStdlib_randx
0x01B3          0x085B          MOVF      FARG_srand_x+1, 0
0x01B4          0x00B6          MOVWF     __Lib_CStdlib_randx+1
0x01B5          0x01B7          CLRF      __Lib_CStdlib_randx+2
0x01B6          0x01B8          CLRF      __Lib_CStdlib_randx+3
;__Lib_CStdlib.c,303 ::
0x01B7          0x3001          MOVLW     1
0x01B8          0x00B3          MOVWF     __Lib_CStdlib_randf
;__Lib_CStdlib.c,304 ::
L_end_srand:
0x01B9          0x0008          RETURN
; end of _srand
_Div_8x8_U:
;__Lib_Math.c,188 ::
;__Lib_Math.c,195 ::
0x01BA          0x0020          MOVLB     0
;__Lib_Math.c,196 ::
0x01BB          0x01F8          CLRF      R8
;__Lib_Math.c,197 ::
0x01BC          0x3008          MOVLW     8
;__Lib_Math.c,198 ::
0x01BD          0x00FC          MOVWF     R12
;__Lib_Math.c,199 ::
0x01BE          0x0D70          RLF       R0, 0
;__Lib_Math.c,200 ::
0x01BF          0x0DF8          RLF       R8, 1
;__Lib_Math.c,201 ::
0x01C0          0x0874          MOVF      R4, 0
;__Lib_Math.c,202 ::
0x01C1          0x02F8          SUBWF     R8, 1
;__Lib_Math.c,203 ::
0x01C2          0x1803          BTFSC     STATUS, 0
;__Lib_Math.c,204 ::
0x01C3          0x29C6          GOTO      $+3
;__Lib_Math.c,205 ::
0x01C4          0x07F8          ADDWF     R8, 1
;__Lib_Math.c,206 ::
0x01C5          0x1003          BCF       STATUS, 0
;__Lib_Math.c,207 ::
0x01C6          0x0DF0          RLF       R0, 1
;__Lib_Math.c,208 ::
```

```
0x01C7          0x0BFC          DECFSZ     R12, 1
;__Lib_Math.c,209 ::
0x01C8          0x29BE          GOTO       $-10
;__Lib_Math.c,211 ::
L_end_Div_8x8_U:
0x01C9          0x0008          RETURN
; end of _Div_8x8_U
_NRM3232:
;__Lib_MathDouble.c,167 ::
;__Lib_MathDouble.c,173 ::
;__Lib_MathDouble.c,175 ::
0x01CA          0x01A0          CLRF       __math_tempbD
;__Lib_MathDouble.c,176 ::
0x01CB          0x0872          MOVF       R2, 0
;__Lib_MathDouble.c,177 ::
0x01CC          0x1D03          BTFSS      STATUS, 2
;__Lib_MathDouble.c,178 ::
0x01CD          0x29DF          GOTO       NORM3232
;__Lib_MathDouble.c,179 ::
0x01CE          0x0871          MOVF       R1, 0
;__Lib_MathDouble.c,180 ::
0x01CF          0x00F2          MOVWF      R2
;__Lib_MathDouble.c,181 ::
0x01D0          0x0870          MOVF       R0, 0
;__Lib_MathDouble.c,182 ::
0x01D1          0x00F1          MOVWF      R1
;__Lib_MathDouble.c,183 ::
0x01D2          0x01F0          CLRF       R0
;__Lib_MathDouble.c,184 ::
0x01D3          0x15A0          BSF        __math_tempbD, 3
;__Lib_MathDouble.c,186 ::
0x01D4          0x0872          MOVF       R2, 0
;__Lib_MathDouble.c,187 ::
0x01D5          0x1D03          BTFSS      STATUS, 2
;__Lib_MathDouble.c,188 ::
0x01D6          0x29DF          GOTO       NORM3232
;__Lib_MathDouble.c,189 ::
0x01D7          0x0871          MOVF       R1, 0
;__Lib_MathDouble.c,190 ::
0x01D8          0x00F2          MOVWF      R2
;__Lib_MathDouble.c,191 ::
0x01D9          0x01F1          CLRF       R1
;__Lib_MathDouble.c,192 ::
0x01DA          0x11A0          BCF        __math_tempbD, 3
;__Lib_MathDouble.c,193 ::
0x01DB          0x1620          BSF        __math_tempbD, 4
;__Lib_MathDouble.c,195 ::
```

```
0x01DC	0x0872	MOVF	R2, 0
;__Lib_MathDouble.c,196 ::
0x01DD	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,197 ::
0x01DE	0x29ED	GOTO	JPNRES032
;__Lib_MathDouble.c,198 ::
NORM3232:
;__Lib_MathDouble.c,199 ::
0x01DF	0x0820	MOVF	__math_tempbD, 0
;__Lib_MathDouble.c,200 ::
0x01E0	0x02F3	SUBWF	R3, 1
;__Lib_MathDouble.c,201 ::
0x01E1	0x1D03	BTFSS	STATUS, 2
;__Lib_MathDouble.c,202 ::
0x01E2	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,203 ::
0x01E3	0x29F1	GOTO	JPNSETFUN32
;__Lib_MathDouble.c,204 ::
0x01E4	0x1003	BCF	STATUS, 0
;__Lib_MathDouble.c,205 ::
NORM3232A:
;__Lib_MathDouble.c,206 ::
0x01E5	0x1BF2	BTFSC	R2, 7
;__Lib_MathDouble.c,207 ::
0x01E6	0x29F5	GOTO	JMPNFIXSIGN32
;__Lib_MathDouble.c,208 ::
0x01E7	0x0DF0	RLF	R0, 1
;__Lib_MathDouble.c,209 ::
0x01E8	0x0DF1	RLF	R1, 1
;__Lib_MathDouble.c,210 ::
0x01E9	0x0DF2	RLF	R2, 1
;__Lib_MathDouble.c,211 ::
0x01EA	0x0BF3	DECFSZ	R3, 1
;__Lib_MathDouble.c,212 ::
0x01EB	0x29E5	GOTO	NORM3232A
;__Lib_MathDouble.c,213 ::
0x01EC	0x29F1	GOTO	JPNSETFUN32
;__Lib_MathDouble.c,214 ::
JPNRES032:
;__Lib_MathDouble.c,215 ::
0x01ED	0x3001	MOVLW	1
;__Lib_MathDouble.c,216 ::
0x01EE	0x00FC	MOVWF	R12
;__Lib_MathDouble.c,218 ::
0x01EF	0x2023	CALL	_RES032
;__Lib_MathDouble.c,220 ::
0x01F0	0x29F8	GOTO	NORM32EEE
```

```
;__Lib_MathDouble.c,221 ::
JPNSETFUN32:
;__Lib_MathDouble.c,222 ::
0x01F1          0x3002          MOVLW     2
;__Lib_MathDouble.c,223 ::
0x01F2          0x00FC          MOVWF     R12
;__Lib_MathDouble.c,225 ::
0x01F3          0x202A          CALL      _SETFUN32
;__Lib_MathDouble.c,227 ::
0x01F4          0x29F8          GOTO      NORM32EEE
;__Lib_MathDouble.c,228 ::
JMPNFIXSIGN32:
;__Lib_MathDouble.c,229 ::
0x01F5          0x3004          MOVLW     4
;__Lib_MathDouble.c,230 ::
0x01F6          0x00FC          MOVWF     R12
;__Lib_MathDouble.c,232 ::
0x01F7          0x2036          CALL      _FIXSIGN32
;__Lib_MathDouble.c,234 ::
NORM32EEE:
;__Lib_MathDouble.c,236 ::
L_end_NRM3232:
0x01F8          0x0008          RETURN
; end of _NRM3232
_FloatToStr:
;__Lib_Conversions.c,472 ::
;__Lib_Conversions.c,474 ::
0x01F9          0x0020          MOVLB     0
0x01FA          0x01D8          CLRF      FloatToStr_bpoint_L0
0x01FB          0x01D9          CLRF      FloatToStr_dexpon_L0
;__Lib_Conversions.c,479 ::
0x01FC          0x084C          MOVF      FARG_FloatToStr_fnum, 0
0x01FD          0x00D4          MOVWF     FloatToStr_un_L0
0x01FE          0x084D          MOVF      FARG_FloatToStr_fnum+1, 0
0x01FF          0x00D5          MOVWF     FloatToStr_un_L0+1
0x0200          0x084E          MOVF      FARG_FloatToStr_fnum+2, 0
0x0201          0x00D6          MOVWF     FloatToStr_un_L0+2
0x0202          0x084F          MOVF      FARG_FloatToStr_fnum+3, 0
0x0203          0x00D7          MOVWF     FloatToStr_un_L0+3
;__Lib_Conversions.c,480 ::
0x0204          0x0857          MOVF      FloatToStr_un_L0+3, 0
0x0205          0x3AFF          XORLW     255
0x0206          0x1D03          BTFSS     STATUS, 2
0x0207          0x2A12          GOTO      L__FloatToStr145
0x0208          0x0856          MOVF      FloatToStr_un_L0+2, 0
0x0209          0x3AFF          XORLW     255
0x020A          0x1D03          BTFSS     STATUS, 2
```

```
0x020B	0x2A12	GOTO	L__FloatToStr145
0x020C	0x0855	MOVF	FloatToStr_un_L0+1, 0
0x020D	0x3AFF	XORLW	255
0x020E	0x1D03	BTFSS	STATUS, 2
0x020F	0x2A12	GOTO	L__FloatToStr145
0x0210	0x0854	MOVF	FloatToStr_un_L0, 0
0x0211	0x3AFF	XORLW	255
L__FloatToStr145:
0x0212	0x1D03	BTFSS	STATUS, 2
0x0213	0x2A20	GOTO	L_FloatToStr76
;__Lib_Conversions.c,481 ::
0x0214	0x0850	MOVF	FARG_FloatToStr_str, 0
0x0215	0x00DA	MOVWF	FARG_strcpy_to
0x0216	0x0851	MOVF	FARG_FloatToStr_str+1, 0
0x0217	0x00DB	MOVWF	FARG_strcpy_to+1
0x0218	0x3024	MOVLW	?lstr1___Lib_Conversions
0x0219	0x00DC	MOVWF	FARG_strcpy_from
0x021A	0x3000	MOVLW	hi_addr(?lstr1___Lib_Conversions)
0x021B	0x00DD	MOVWF	FARG_strcpy_from+1
0x021C	0x2089	CALL	_strcpy
;__Lib_Conversions.c,482 ::
0x021D	0x3003	MOVLW	3
0x021E	0x00F0	MOVWF	R0
0x021F	0x2C1E	GOTO	L_end_FloatToStr
;__Lib_Conversions.c,483 ::
L_FloatToStr76:
;__Lib_Conversions.c,484 ::
0x0220	0x3001	MOVLW	1
0x0221	0x00D2	MOVWF	FloatToStr_i_L0
;__Lib_Conversions.c,485 ::
0x0222	0x1FD6	BTFSS	FloatToStr_un_L0+2, 7
0x0223	0x2A33	GOTO	L_FloatToStr77
;__Lib_Conversions.c,486 ::
0x0224	0x3080	MOVLW	128
0x0225	0x0656	XORWF	FloatToStr_un_L0+2, 0
0x0226	0x00F0	MOVWF	R0
0x0227	0x0870	MOVF	R0, 0
0x0228	0x00D6	MOVWF	FloatToStr_un_L0+2
;__Lib_Conversions.c,487 ::
0x0229	0x0AD2	INCF	FloatToStr_i_L0, 1
;__Lib_Conversions.c,488 ::
0x022A	0x0850	MOVF	FARG_FloatToStr_str, 0
0x022B	0x0086	MOVWF	FSR1
0x022C	0x0851	MOVF	FARG_FloatToStr_str+1, 0
0x022D	0x0087	MOVWF	FSR1H
0x022E	0x302D	MOVLW	45
0x022F	0x0081	MOVWF	INDF1
```

```
0x0230          0x0AD0          INCF       FARG_FloatToStr_str, 1
0x0231          0x1903          BTFSC      STATUS, 2
0x0232          0x0AD1          INCF       FARG_FloatToStr_str+1, 1
;__Lib_Conversions.c,489 ::
L_FloatToStr77:
;__Lib_Conversions.c,490 ::
0x0233          0x3000          MOVLW      0
0x0234          0x00F0          MOVWF      R0
0x0235          0x0657          XORWF      FloatToStr_un_L0+3, 0
0x0236          0x1D03          BTFSS      STATUS, 2
0x0237          0x2A42          GOTO       L__FloatToStr146
0x0238          0x0870          MOVF       R0, 0
0x0239          0x0656          XORWF      FloatToStr_un_L0+2, 0
0x023A          0x1D03          BTFSS      STATUS, 2
0x023B          0x2A42          GOTO       L__FloatToStr146
0x023C          0x0870          MOVF       R0, 0
0x023D          0x0655          XORWF      FloatToStr_un_L0+1, 0
0x023E          0x1D03          BTFSS      STATUS, 2
0x023F          0x2A42          GOTO       L__FloatToStr146
0x0240          0x0854          MOVF       FloatToStr_un_L0, 0
0x0241          0x3A00          XORLW      0
L__FloatToStr146:
0x0242          0x1D03          BTFSS      STATUS, 2
0x0243          0x2A4F          GOTO       L_FloatToStr78
;__Lib_Conversions.c,491 ::
0x0244          0x0850          MOVF       FARG_FloatToStr_str, 0
0x0245          0x00DA          MOVWF      FARG_strcpy_to
0x0246          0x0851          MOVF       FARG_FloatToStr_str+1, 0
0x0247          0x00DB          MOVWF      FARG_strcpy_to+1
0x0248          0x3022          MOVLW      ?lstr2___Lib_Conversions
0x0249          0x00DC          MOVWF      FARG_strcpy_from
0x024A          0x3000          MOVLW      hi_addr(?lstr2___Lib_Conversions)
0x024B          0x00DD          MOVWF      FARG_strcpy_from+1
0x024C          0x2089          CALL       _strcpy
;__Lib_Conversions.c,492 ::
0x024D          0x01F0          CLRF       R0
0x024E          0x2C1E          GOTO       L_end_FloatToStr
;__Lib_Conversions.c,493 ::
L_FloatToStr78:
;__Lib_Conversions.c,494 ::
0x024F          0x0857          MOVF       FloatToStr_un_L0+3, 0
0x0250          0x3AFF          XORLW      255
0x0251          0x1D03          BTFSS      STATUS, 2
0x0252          0x2A5D          GOTO       L__FloatToStr147
0x0253          0x0856          MOVF       FloatToStr_un_L0+2, 0
0x0254          0x3A00          XORLW      0
0x0255          0x1D03          BTFSS      STATUS, 2
```

```
0x0256  0x2A5D  GOTO    L__FloatToStr147
0x0257  0x0855  MOVF    FloatToStr_un_L0+1, 0
0x0258  0x3A00  XORLW   0
0x0259  0x1D03  BTFSS   STATUS, 2
0x025A  0x2A5D  GOTO    L__FloatToStr147
0x025B  0x0854  MOVF    FloatToStr_un_L0, 0
0x025C  0x3A00  XORLW   0
L__FloatToStr147:
0x025D  0x1D03  BTFSS   STATUS, 2
0x025E  0x2A6B  GOTO    L_FloatToStr79
;__Lib_Conversions.c,495 ::
0x025F  0x0850  MOVF    FARG_FloatToStr_str, 0
0x0260  0x00DA  MOVWF   FARG_strcpy_to
0x0261  0x0851  MOVF    FARG_FloatToStr_str+1, 0
0x0262  0x00DB  MOVWF   FARG_strcpy_to+1
0x0263  0x3028  MOVLW   ?lstr3___Lib_Conversions
0x0264  0x00DC  MOVWF   FARG_strcpy_from
0x0265  0x3000  MOVLW   hi_addr(?lstr3___Lib_Conversions)
0x0266  0x00DD  MOVWF   FARG_strcpy_from+1
0x0267  0x2089  CALL    _strcpy
;__Lib_Conversions.c,496 ::
0x0268  0x0852  MOVF    FloatToStr_i_L0, 0
0x0269  0x00F0  MOVWF   R0
0x026A  0x2C1E  GOTO    L_end_FloatToStr
;__Lib_Conversions.c,497 ::
L_FloatToStr79:
;__Lib_Conversions.c,505 ::
L_FloatToStr80:
0x026B  0x3000  MOVLW   0
0x026C  0x00F4  MOVWF   R4
0x026D  0x3000  MOVLW   0
0x026E  0x00F5  MOVWF   R5
0x026F  0x3000  MOVLW   0
0x0270  0x00F6  MOVWF   R6
0x0271  0x307F  MOVLW   127
0x0272  0x00F7  MOVWF   R7
0x0273  0x0854  MOVF    FloatToStr_un_L0, 0
0x0274  0x00F0  MOVWF   R0
0x0275  0x0855  MOVF    FloatToStr_un_L0+1, 0
0x0276  0x00F1  MOVWF   R1
0x0277  0x0856  MOVF    FloatToStr_un_L0+2, 0
0x0278  0x00F2  MOVWF   R2
0x0279  0x0857  MOVF    FloatToStr_un_L0+3, 0
0x027A  0x00F3  MOVWF   R3
0x027B  0x203A  CALL    _Compare_Double
0x027C  0x3001  MOVLW   1
0x027D  0x1803  BTFSC   STATUS, 0
```

```
0x027E          0x3000          MOVLW     0
0x027F          0x00F0          MOVWF     R0
0x0280          0x0870          MOVF      R0, 0
0x0281          0x1903          BTFSC     STATUS, 2
0x0282          0x2A9E          GOTO      L_FloatToStr81
;__Lib_Conversions.c,506 ::
0x0283          0x0854          MOVF      FloatToStr_un_L0, 0
0x0284          0x00F0          MOVWF     R0
0x0285          0x0855          MOVF      FloatToStr_un_L0+1, 0
0x0286          0x00F1          MOVWF     R1
0x0287          0x0856          MOVF      FloatToStr_un_L0+2, 0
0x0288          0x00F2          MOVWF     R2
0x0289          0x0857          MOVF      FloatToStr_un_L0+3, 0
0x028A          0x00F3          MOVWF     R3
0x028B          0x3000          MOVLW     0
0x028C          0x00F4          MOVWF     R4
0x028D          0x3000          MOVLW     0
0x028E          0x00F5          MOVWF     R5
0x028F          0x3020          MOVLW     32
0x0290          0x00F6          MOVWF     R6
0x0291          0x3082          MOVLW     130
0x0292          0x00F7          MOVWF     R7
0x0293          0x20C0          CALL      _Mul_32x32_FP
0x0294          0x0870          MOVF      R0, 0
0x0295          0x00D4          MOVWF     FloatToStr_un_L0
0x0296          0x0871          MOVF      R1, 0
0x0297          0x00D5          MOVWF     FloatToStr_un_L0+1
0x0298          0x0872          MOVF      R2, 0
0x0299          0x00D6          MOVWF     FloatToStr_un_L0+2
0x029A          0x0873          MOVF      R3, 0
0x029B          0x00D7          MOVWF     FloatToStr_un_L0+3
;__Lib_Conversions.c,507 ::
0x029C          0x03D9          DECF      FloatToStr_dexpon_L0, 1
;__Lib_Conversions.c,508 ::
0x029D          0x2A6B          GOTO      L_FloatToStr80
L_FloatToStr81:
;__Lib_Conversions.c,513 ::
L_FloatToStr82:
0x029E          0x3000          MOVLW     0
0x029F          0x00F4          MOVWF     R4
0x02A0          0x3000          MOVLW     0
0x02A1          0x00F5          MOVWF     R5
0x02A2          0x3020          MOVLW     32
0x02A3          0x00F6          MOVWF     R6
0x02A4          0x3082          MOVLW     130
0x02A5          0x00F7          MOVWF     R7
0x02A6          0x0854          MOVF      FloatToStr_un_L0, 0
```

```
0x02A7  0x00F0  MOVWF   R0
0x02A8  0x0855  MOVF    FloatToStr_un_L0+1, 0
0x02A9  0x00F1  MOVWF   R1
0x02AA  0x0856  MOVF    FloatToStr_un_L0+2, 0
0x02AB  0x00F2  MOVWF   R2
0x02AC  0x0857  MOVF    FloatToStr_un_L0+3, 0
0x02AD  0x00F3  MOVWF   R3
0x02AE  0x203A  CALL    _Compare_Double
0x02AF  0x3001  MOVLW   1
0x02B0  0x1C03  BTFSS   STATUS, 0
0x02B1  0x3000  MOVLW   0
0x02B2  0x00F0  MOVWF   R0
0x02B3  0x0870  MOVF    R0, 0
0x02B4  0x1903  BTFSC   STATUS, 2
0x02B5  0x2AD1  GOTO    L_FloatToStr83
;__Lib_Conversions.c,514 ::
0x02B6  0x0854  MOVF    FloatToStr_un_L0, 0
0x02B7  0x00F0  MOVWF   R0
0x02B8  0x0855  MOVF    FloatToStr_un_L0+1, 0
0x02B9  0x00F1  MOVWF   R1
0x02BA  0x0856  MOVF    FloatToStr_un_L0+2, 0
0x02BB  0x00F2  MOVWF   R2
0x02BC  0x0857  MOVF    FloatToStr_un_L0+3, 0
0x02BD  0x00F3  MOVWF   R3
0x02BE  0x30CD  MOVLW   205
0x02BF  0x00F4  MOVWF   R4
0x02C0  0x30CC  MOVLW   204
0x02C1  0x00F5  MOVWF   R5
0x02C2  0x304C  MOVLW   76
0x02C3  0x00F6  MOVWF   R6
0x02C4  0x307B  MOVLW   123
0x02C5  0x00F7  MOVWF   R7
0x02C6  0x20C0  CALL    _Mul_32x32_FP
0x02C7  0x0870  MOVF    R0, 0
0x02C8  0x00D4  MOVWF   FloatToStr_un_L0
0x02C9  0x0871  MOVF    R1, 0
0x02CA  0x00D5  MOVWF   FloatToStr_un_L0+1
0x02CB  0x0872  MOVF    R2, 0
0x02CC  0x00D6  MOVWF   FloatToStr_un_L0+2
0x02CD  0x0873  MOVF    R3, 0
0x02CE  0x00D7  MOVWF   FloatToStr_un_L0+3
;__Lib_Conversions.c,515 ::
0x02CF  0x0AD9  INCF    FloatToStr_dexpon_L0, 1
;__Lib_Conversions.c,516 ::
0x02D0  0x2A9E  GOTO    L_FloatToStr82
L_FloatToStr83:
;__Lib_Conversions.c,525 ::
```

```
0x02D1	0x30FF	MOVLW	255
0x02D2	0x0554	ANDWF	FloatToStr_un_L0, 0
0x02D3	0x00F8	MOVWF	R8
0x02D4	0x30FF	MOVLW	255
0x02D5	0x0555	ANDWF	FloatToStr_un_L0+1, 0
0x02D6	0x00F9	MOVWF	R9
0x02D7	0x307F	MOVLW	127
0x02D8	0x0556	ANDWF	FloatToStr_un_L0+2, 0
0x02D9	0x00FA	MOVWF	R10
0x02DA	0x3000	MOVLW	0
0x02DB	0x0557	ANDWF	FloatToStr_un_L0+3, 0
0x02DC	0x00FB	MOVWF	R11
;__Lib_Conversions.c,526 ::
0x02DD	0x0878	MOVF	R8, 0
0x02DE	0x00F4	MOVWF	R4
0x02DF	0x0879	MOVF	R9, 0
0x02E0	0x00F5	MOVWF	R5
0x02E1	0x087A	MOVF	R10, 0
0x02E2	0x00F6	MOVWF	R6
0x02E3	0x087B	MOVF	R11, 0
0x02E4	0x00F7	MOVWF	R7
0x02E5	0x35F4	LSLF	R4, 1
0x02E6	0x0DF5	RLF	R5, 1
0x02E7	0x0DF6	RLF	R6, 1
0x02E8	0x0DF7	RLF	R7, 1
;__Lib_Conversions.c,527 ::
0x02E9	0x3000	MOVLW	0
0x02EA	0x0554	ANDWF	FloatToStr_un_L0, 0
0x02EB	0x00F0	MOVWF	R0
0x02EC	0x3000	MOVLW	0
0x02ED	0x0555	ANDWF	FloatToStr_un_L0+1, 0
0x02EE	0x00F1	MOVWF	R1
0x02EF	0x3000	MOVLW	0
0x02F0	0x0556	ANDWF	FloatToStr_un_L0+2, 0
0x02F1	0x00F2	MOVWF	R2
0x02F2	0x30FF	MOVLW	255
0x02F3	0x0557	ANDWF	FloatToStr_un_L0+3, 0
0x02F4	0x00F3	MOVWF	R3
0x02F5	0x0870	MOVF	R0, 0
0x02F6	0x00D4	MOVWF	FloatToStr_un_L0
0x02F7	0x0871	MOVF	R1, 0
0x02F8	0x00D5	MOVWF	FloatToStr_un_L0+1
0x02F9	0x0872	MOVF	R2, 0
0x02FA	0x00D6	MOVWF	FloatToStr_un_L0+2
0x02FB	0x0873	MOVF	R3, 0
0x02FC	0x00D7	MOVWF	FloatToStr_un_L0+3
;__Lib_Conversions.c,528 ::
```

```
0x02FD  0x0874  MOVF    R4, 0
0x02FE  0x0454  IORWF   FloatToStr_un_L0, 0
0x02FF  0x00F0  MOVWF   R0
0x0300  0x0875  MOVF    R5, 0
0x0301  0x0455  IORWF   FloatToStr_un_L0+1, 0
0x0302  0x00F1  MOVWF   R1
0x0303  0x0876  MOVF    R6, 0
0x0304  0x0456  IORWF   FloatToStr_un_L0+2, 0
0x0305  0x00F2  MOVWF   R2
0x0306  0x0877  MOVF    R7, 0
0x0307  0x0457  IORWF   FloatToStr_un_L0+3, 0
0x0308  0x00F3  MOVWF   R3
0x0309  0x0870  MOVF    R0, 0
0x030A  0x00D4  MOVWF   FloatToStr_un_L0
0x030B  0x0871  MOVF    R1, 0
0x030C  0x00D5  MOVWF   FloatToStr_un_L0+1
0x030D  0x0872  MOVF    R2, 0
0x030E  0x00D6  MOVWF   FloatToStr_un_L0+2
0x030F  0x0873  MOVF    R3, 0
0x0310  0x00D7  MOVWF   FloatToStr_un_L0+3
;__Lib_Conversions.c,530 ::
0x0311  0x307F  MOVLW   127
0x0312  0x0257  SUBWF   FloatToStr_un_L0+3, 0
0x0313  0x00F0  MOVWF   R0
0x0314  0x0870  MOVF    R0, 0
0x0315  0x00D3  MOVWF   FloatToStr_d_L0
;__Lib_Conversions.c,533 ::
0x0316  0x3001  MOVLW   1
0x0317  0x00D7  MOVWF   FloatToStr_un_L0+3
;__Lib_Conversions.c,534 ::
0x0318  0x0870  MOVF    R0, 0
0x0319  0x00F4  MOVWF   R4
0x031A  0x0854  MOVF    FloatToStr_un_L0, 0
0x031B  0x00F0  MOVWF   R0
0x031C  0x0855  MOVF    FloatToStr_un_L0+1, 0
0x031D  0x00F1  MOVWF   R1
0x031E  0x0856  MOVF    FloatToStr_un_L0+2, 0
0x031F  0x00F2  MOVWF   R2
0x0320  0x0857  MOVF    FloatToStr_un_L0+3, 0
0x0321  0x00F3  MOVWF   R3
0x0322  0x0874  MOVF    R4, 0
L__FloatToStr148:
0x0323  0x1903  BTFSC   STATUS, 2
0x0324  0x2B2B  GOTO    L__FloatToStr149
0x0325  0x35F0  LSLF    R0, 1
0x0326  0x0DF1  RLF     R1, 1
0x0327  0x0DF2  RLF     R2, 1
```

```
0x0328          0x0DF3          RLF       R3, 1
0x0329          0x3EFF          ADDLW     255
0x032A          0x2B23          GOTO      L__FloatToStr148
L__FloatToStr149:
0x032B          0x0870          MOVF      R0, 0
0x032C          0x00D4          MOVWF     FloatToStr_un_L0
0x032D          0x0871          MOVF      R1, 0
0x032E          0x00D5          MOVWF     FloatToStr_un_L0+1
0x032F          0x0872          MOVF      R2, 0
0x0330          0x00D6          MOVWF     FloatToStr_un_L0+2
0x0331          0x0873          MOVF      R3, 0
0x0332          0x00D7          MOVWF     FloatToStr_un_L0+3
;__Lib_Conversions.c,535 ::
0x0333          0x0850          MOVF      FARG_FloatToStr_str, 0
0x0334          0x0086          MOVWF     FSR1
0x0335          0x0851          MOVF      FARG_FloatToStr_str+1, 0
0x0336          0x0087          MOVWF     FSR1H
0x0337          0x0857          MOVF      FloatToStr_un_L0+3, 0
0x0338          0x3E30          ADDLW     48
0x0339          0x0081          MOVWF     INDF1
0x033A          0x0AD0          INCF      FARG_FloatToStr_str, 1
0x033B          0x1903          BTFSC     STATUS, 2
0x033C          0x0AD1          INCF      FARG_FloatToStr_str+1, 1
;__Lib_Conversions.c,536 ::
0x033D          0x3080          MOVLW     128
0x033E          0x0659          XORWF     FloatToStr_dexpon_L0, 0
0x033F          0x00F0          MOVWF     R0
0x0340          0x3080          MOVLW     128
0x0341          0x3A01          XORLW     1
0x0342          0x0270          SUBWF     R0, 0
0x0343          0x1C03          BTFSS     STATUS, 0
0x0344          0x2B4E          GOTO      L__FloatToStr100
0x0345          0x3080          MOVLW     128
0x0346          0x3A06          XORLW     6
0x0347          0x00F0          MOVWF     R0
0x0348          0x3080          MOVLW     128
0x0349          0x0659          XORWF     FloatToStr_dexpon_L0, 0
0x034A          0x0270          SUBWF     R0, 0
0x034B          0x1C03          BTFSS     STATUS, 0
0x034C          0x2B4E          GOTO      L__FloatToStr100
0x034D          0x2B59          GOTO      L_FloatToStr86
L__FloatToStr100:
;__Lib_Conversions.c,537 ::
0x034E          0x0850          MOVF      FARG_FloatToStr_str, 0
0x034F          0x0086          MOVWF     FSR1
0x0350          0x0851          MOVF      FARG_FloatToStr_str+1, 0
0x0351          0x0087          MOVWF     FSR1H
```

```
0x0352          0x302E          MOVLW      46
0x0353          0x0081          MOVWF      INDF1
0x0354          0x0AD0          INCF       FARG_FloatToStr_str, 1
0x0355          0x1903          BTFSC      STATUS, 2
0x0356          0x0AD1          INCF       FARG_FloatToStr_str+1, 1
;__Lib_Conversions.c,538 ::
0x0357          0x3001          MOVLW      1
0x0358          0x00D8          MOVWF      FloatToStr_bpoint_L0
;__Lib_Conversions.c,539 ::
L_FloatToStr86:
;__Lib_Conversions.c,540 ::
0x0359          0x3006          MOVLW      6
0x035A          0x00D3          MOVWF      FloatToStr_d_L0
L_FloatToStr87:
0x035B          0x0853          MOVF       FloatToStr_d_L0, 0
0x035C          0x3A00          XORLW      0
0x035D          0x1903          BTFSC      STATUS, 2
0x035E          0x2BB4          GOTO       L_FloatToStr88
;__Lib_Conversions.c,541 ::
0x035F          0x01D7          CLRF       FloatToStr_un_L0+3
;__Lib_Conversions.c,542 ::
0x0360          0x0854          MOVF       FloatToStr_un_L0, 0
0x0361          0x00F0          MOVWF      R0
0x0362          0x0855          MOVF       FloatToStr_un_L0+1, 0
0x0363          0x00F1          MOVWF      R1
0x0364          0x0856          MOVF       FloatToStr_un_L0+2, 0
0x0365          0x00F2          MOVWF      R2
0x0366          0x0857          MOVF       FloatToStr_un_L0+3, 0
0x0367          0x00F3          MOVWF      R3
0x0368          0x35F0          LSLF       R0, 1
0x0369          0x0DF1          RLF        R1, 1
0x036A          0x0DF2          RLF        R2, 1
0x036B          0x0DF3          RLF        R3, 1
0x036C          0x35F0          LSLF       R0, 1
0x036D          0x0DF1          RLF        R1, 1
0x036E          0x0DF2          RLF        R2, 1
0x036F          0x0DF3          RLF        R3, 1
0x0370          0x0854          MOVF       FloatToStr_un_L0, 0
0x0371          0x07F0          ADDWF      R0, 1
0x0372          0x0855          MOVF       FloatToStr_un_L0+1, 0
0x0373          0x3DF1          ADDWFC     R1, 1
0x0374          0x0856          MOVF       FloatToStr_un_L0+2, 0
0x0375          0x3DF2          ADDWFC     R2, 1
0x0376          0x0857          MOVF       FloatToStr_un_L0+3, 0
0x0377          0x3DF3          ADDWFC     R3, 1
0x0378          0x0870          MOVF       R0, 0
0x0379          0x00D4          MOVWF      FloatToStr_un_L0
```

```
0x037A  0x0871  MOVF    R1, 0
0x037B  0x00D5  MOVWF   FloatToStr_un_L0+1
0x037C  0x0872  MOVF    R2, 0
0x037D  0x00D6  MOVWF   FloatToStr_un_L0+2
0x037E  0x0873  MOVF    R3, 0
0x037F  0x00D7  MOVWF   FloatToStr_un_L0+3
;__Lib_Conversions.c,543 ::
0x0380  0x0854  MOVF    FloatToStr_un_L0, 0
0x0381  0x00F0  MOVWF   R0
0x0382  0x0855  MOVF    FloatToStr_un_L0+1, 0
0x0383  0x00F1  MOVWF   R1
0x0384  0x0856  MOVF    FloatToStr_un_L0+2, 0
0x0385  0x00F2  MOVWF   R2
0x0386  0x0857  MOVF    FloatToStr_un_L0+3, 0
0x0387  0x00F3  MOVWF   R3
0x0388  0x35F0  LSLF    R0, 1
0x0389  0x0DF1  RLF     R1, 1
0x038A  0x0DF2  RLF     R2, 1
0x038B  0x0DF3  RLF     R3, 1
0x038C  0x0870  MOVF    R0, 0
0x038D  0x00D4  MOVWF   FloatToStr_un_L0
0x038E  0x0871  MOVF    R1, 0
0x038F  0x00D5  MOVWF   FloatToStr_un_L0+1
0x0390  0x0872  MOVF    R2, 0
0x0391  0x00D6  MOVWF   FloatToStr_un_L0+2
0x0392  0x0873  MOVF    R3, 0
0x0393  0x00D7  MOVWF   FloatToStr_un_L0+3
;__Lib_Conversions.c,544 ::
0x0394  0x0850  MOVF    FARG_FloatToStr_str, 0
0x0395  0x0086  MOVWF   FSR1
0x0396  0x0851  MOVF    FARG_FloatToStr_str+1, 0
0x0397  0x0087  MOVWF   FSR1H
0x0398  0x0857  MOVF    FloatToStr_un_L0+3, 0
0x0399  0x3E30  ADDLW   48
0x039A  0x0081  MOVWF   INDF1
0x039B  0x0AD0  INCF    FARG_FloatToStr_str, 1
0x039C  0x1903  BTFSC   STATUS, 2
0x039D  0x0AD1  INCF    FARG_FloatToStr_str+1, 1
;__Lib_Conversions.c,545 ::
0x039E  0x0858  MOVF    FloatToStr_bpoint_L0, 0
0x039F  0x3A00  XORLW   0
0x03A0  0x1D03  BTFSS   STATUS, 2
0x03A1  0x2BB2  GOTO    L_FloatToStr90
;__Lib_Conversions.c,546 ::
0x03A2  0x03D9  DECF    FloatToStr_dexpon_L0, 1
0x03A3  0x0859  MOVF    FloatToStr_dexpon_L0, 0
0x03A4  0x3A00  XORLW   0
```

```
0x03A5  0x1D03  BTFSS   STATUS, 2
0x03A6  0x2BB2  GOTO    L_FloatToStr91
;__Lib_Conversions.c,547 ::
0x03A7  0x0850  MOVF    FARG_FloatToStr_str, 0
0x03A8  0x0086  MOVWF   FSR1
0x03A9  0x0851  MOVF    FARG_FloatToStr_str+1, 0
0x03AA  0x0087  MOVWF   FSR1H
0x03AB  0x302E  MOVLW   46
0x03AC  0x0081  MOVWF   INDF1
0x03AD  0x0AD0  INCF    FARG_FloatToStr_str, 1
0x03AE  0x1903  BTFSC   STATUS, 2
0x03AF  0x0AD1  INCF    FARG_FloatToStr_str+1, 1
;__Lib_Conversions.c,548 ::
0x03B0  0x3001  MOVLW   1
0x03B1  0x00D8  MOVWF   FloatToStr_bpoint_L0
;__Lib_Conversions.c,549 ::
L_FloatToStr91:
L_FloatToStr90:
;__Lib_Conversions.c,540 ::
0x03B2  0x03D3  DECF    FloatToStr_d_L0, 1
;__Lib_Conversions.c,550 ::
0x03B3  0x2B5B  GOTO    L_FloatToStr87
L_FloatToStr88:
;__Lib_Conversions.c,551 ::
L_FloatToStr92:
0x03B4  0x30FF  MOVLW   255
0x03B5  0x0750  ADDWF   FARG_FloatToStr_str, 0
0x03B6  0x0084  MOVWF   FSR0
0x03B7  0x30FF  MOVLW   255
0x03B8  0x3D51  ADDWFC  FARG_FloatToStr_str+1, 0
0x03B9  0x0085  MOVWF   FSR0H
0x03BA  0x0800  MOVF    INDF0, 0
0x03BB  0x3A30  XORLW   48
0x03BC  0x1D03  BTFSS   STATUS, 2
0x03BD  0x2BC3  GOTO    L_FloatToStr93
;__Lib_Conversions.c,552 ::
0x03BE  0x3001  MOVLW   1
0x03BF  0x02D0  SUBWF   FARG_FloatToStr_str, 1
0x03C0  0x3000  MOVLW   0
0x03C1  0x3BD1  SUBWFB  FARG_FloatToStr_str+1, 1
0x03C2  0x2BB4  GOTO    L_FloatToStr92
L_FloatToStr93:
;__Lib_Conversions.c,553 ::
0x03C3  0x30FF  MOVLW   255
0x03C4  0x0750  ADDWF   FARG_FloatToStr_str, 0
0x03C5  0x0084  MOVWF   FSR0
0x03C6  0x30FF  MOVLW   255
```

```
0x03C7          0x3D51          ADDWFC     FARG_FloatToStr_str+1, 0
0x03C8          0x0085          MOVWF      FSR0H
0x03C9          0x0800          MOVF       INDF0, 0
0x03CA          0x3A2E          XORLW      46
0x03CB          0x1D03          BTFSS      STATUS, 2
0x03CC          0x2BD1          GOTO       L_FloatToStr94
;__Lib_Conversions.c,554 ::
0x03CD          0x3001          MOVLW      1
0x03CE          0x02D0          SUBWF      FARG_FloatToStr_str, 1
0x03CF          0x3000          MOVLW      0
0x03D0          0x3BD1          SUBWFB     FARG_FloatToStr_str+1, 1
L_FloatToStr94:
;__Lib_Conversions.c,555 ::
0x03D1          0x0859          MOVF       FloatToStr_dexpon_L0, 0
0x03D2          0x3A00          XORLW      0
0x03D3          0x1903          BTFSC      STATUS, 2
0x03D4          0x2C18          GOTO       L_FloatToStr95
;__Lib_Conversions.c,556 ::
0x03D5          0x0850          MOVF       FARG_FloatToStr_str, 0
0x03D6          0x0086          MOVWF      FSR1
0x03D7          0x0851          MOVF       FARG_FloatToStr_str+1, 0
0x03D8          0x0087          MOVWF      FSR1H
0x03D9          0x3065          MOVLW      101
0x03DA          0x0081          MOVWF      INDF1
0x03DB          0x0AD0          INCF       FARG_FloatToStr_str, 1
0x03DC          0x1903          BTFSC      STATUS, 2
0x03DD          0x0AD1          INCF       FARG_FloatToStr_str+1, 1
;__Lib_Conversions.c,557 ::
0x03DE          0x3080          MOVLW      128
0x03DF          0x0659          XORWF      FloatToStr_dexpon_L0, 0
0x03E0          0x00F0          MOVWF      R0
0x03E1          0x3080          MOVLW      128
0x03E2          0x3A00          XORLW      0
0x03E3          0x0270          SUBWF      R0, 0
0x03E4          0x1803          BTFSC      STATUS, 0
0x03E5          0x2BF2          GOTO       L_FloatToStr96
;__Lib_Conversions.c,558 ::
0x03E6          0x0850          MOVF       FARG_FloatToStr_str, 0
0x03E7          0x0086          MOVWF      FSR1
0x03E8          0x0851          MOVF       FARG_FloatToStr_str+1, 0
0x03E9          0x0087          MOVWF      FSR1H
0x03EA          0x302D          MOVLW      45
0x03EB          0x0081          MOVWF      INDF1
0x03EC          0x0AD0          INCF       FARG_FloatToStr_str, 1
0x03ED          0x1903          BTFSC      STATUS, 2
0x03EE          0x0AD1          INCF       FARG_FloatToStr_str+1, 1
;__Lib_Conversions.c,559 ::
```

```
0x03EF  0x0859  MOVF    FloatToStr_dexpon_L0, 0
0x03F0  0x3C00  SUBLW   0
0x03F1  0x00D9  MOVWF   FloatToStr_dexpon_L0
;__Lib_Conversions.c,560 ::
L_FloatToStr96:
;__Lib_Conversions.c,561 ::
0x03F2  0x0859  MOVF    FloatToStr_dexpon_L0, 0
0x03F3  0x00D3  MOVWF   FloatToStr_d_L0
;__Lib_Conversions.c,562 ::
0x03F4  0x0859  MOVF    FloatToStr_dexpon_L0, 0
0x03F5  0x3C09  SUBLW   9
0x03F6  0x1803  BTFSC   STATUS, 0
0x03F7  0x2C07  GOTO    L_FloatToStr97
;__Lib_Conversions.c,563 ::
0x03F8  0x300A  MOVLW   10
0x03F9  0x00F4  MOVWF   R4
0x03FA  0x0853  MOVF    FloatToStr_d_L0, 0
0x03FB  0x00F0  MOVWF   R0
0x03FC  0x21BA  CALL    _Div_8x8_U
0x03FD  0x0850  MOVF    FARG_FloatToStr_str, 0
0x03FE  0x0086  MOVWF   FSR1
0x03FF  0x0851  MOVF    FARG_FloatToStr_str+1, 0
0x0400  0x0087  MOVWF   FSR1H
0x0401  0x0870  MOVF    R0, 0
0x0402  0x3E30  ADDLW   48
0x0403  0x0081  MOVWF   INDF1
0x0404  0x0AD0  INCF    FARG_FloatToStr_str, 1
0x0405  0x1903  BTFSC   STATUS, 2
0x0406  0x0AD1  INCF    FARG_FloatToStr_str+1, 1
L_FloatToStr97:
;__Lib_Conversions.c,564 ::
0x0407  0x300A  MOVLW   10
0x0408  0x00F4  MOVWF   R4
0x0409  0x0853  MOVF    FloatToStr_d_L0, 0
0x040A  0x00F0  MOVWF   R0
0x040B  0x21BA  CALL    _Div_8x8_U
0x040C  0x0878  MOVF    R8, 0
0x040D  0x00F0  MOVWF   R0
0x040E  0x0850  MOVF    FARG_FloatToStr_str, 0
0x040F  0x0086  MOVWF   FSR1
0x0410  0x0851  MOVF    FARG_FloatToStr_str+1, 0
0x0411  0x0087  MOVWF   FSR1H
0x0412  0x0870  MOVF    R0, 0
0x0413  0x3E30  ADDLW   48
0x0414  0x0081  MOVWF   INDF1
0x0415  0x0AD0  INCF    FARG_FloatToStr_str, 1
0x0416  0x1903  BTFSC   STATUS, 2
```

```
0x0417          0x0AD1          INCF       FARG_FloatToStr_str+1, 1
;__Lib_Conversions.c,565 ::
L_FloatToStr95:
;__Lib_Conversions.c,566 ::
0x0418          0x0850          MOVF       FARG_FloatToStr_str, 0
0x0419          0x0086          MOVWF      FSR1
0x041A          0x0851          MOVF       FARG_FloatToStr_str+1, 0
0x041B          0x0087          MOVWF      FSR1H
0x041C          0x0181          CLRF       INDF1
;__Lib_Conversions.c,567 ::
0x041D          0x01F0          CLRF       R0
;__Lib_Conversions.c,568 ::
L_end_FloatToStr:
0x041E          0x0008          RETURN
; end of _FloatToStr
_rand:
;__Lib_CStdlib.c,307 ::
;__Lib_CStdlib.c,308 ::
0x041F          0x0020          MOVLB      0
0x0420          0x0833          MOVF       __Lib_CStdlib_randf, 0
0x0421          0x1D03          BTFSS      STATUS, 2
0x0422          0x2C28          GOTO       L_rand75
;__Lib_CStdlib.c,309 ::
0x0423          0x3001          MOVLW      1
0x0424          0x00DA          MOVWF      FARG_srand_x
0x0425          0x3000          MOVLW      0
0x0426          0x00DB          MOVWF      FARG_srand_x+1
0x0427          0x21B0          CALL       _srand
L_rand75:
;__Lib_CStdlib.c,310 ::
0x0428          0x0835          MOVF       __Lib_CStdlib_randx, 0
0x0429          0x00F0          MOVWF      R0
0x042A          0x0836          MOVF       __Lib_CStdlib_randx+1, 0
0x042B          0x00F1          MOVWF      R1
0x042C          0x0837          MOVF       __Lib_CStdlib_randx+2, 0
0x042D          0x00F2          MOVWF      R2
0x042E          0x0838          MOVF       __Lib_CStdlib_randx+3, 0
0x042F          0x00F3          MOVWF      R3
0x0430          0x306D          MOVLW      109
0x0431          0x00F4          MOVWF      R4
0x0432          0x304E          MOVLW      78
0x0433          0x00F5          MOVWF      R5
0x0434          0x30C6          MOVLW      198
0x0435          0x00F6          MOVWF      R6
0x0436          0x3041          MOVLW      65
0x0437          0x00F7          MOVWF      R7
0x0438          0x217A          CALL       _Mul_32x32_U
```

```
0x0439      0x3039      MOVLW    57
0x043A      0x0770      ADDWF    R0, 0
0x043B      0x00F5      MOVWF    R5
0x043C      0x3030      MOVLW    48
0x043D      0x3D71      ADDWFC   R1, 0
0x043E      0x00F6      MOVWF    R6
0x043F      0x3000      MOVLW    0
0x0440      0x3D72      ADDWFC   R2, 0
0x0441      0x00F7      MOVWF    R7
0x0442      0x3000      MOVLW    0
0x0443      0x3D73      ADDWFC   R3, 0
0x0444      0x00F8      MOVWF    R8
0x0445      0x0877      MOVF     R7, 0
0x0446      0x00F0      MOVWF    R0
0x0447      0x0878      MOVF     R8, 0
0x0448      0x00F1      MOVWF    R1
0x0449      0x3000      MOVLW    0
0x044A      0x1BF8      BTFSC    R8, 7
0x044B      0x30FF      MOVLW    255
0x044C      0x00F3      MOVWF    R3
0x044D      0x30FF      MOVLW    255
0x044E      0x05F0      ANDWF    R0, 1
0x044F      0x307F      MOVLW    127
0x0450      0x05F1      ANDWF    R1, 1
0x0451      0x3000      MOVLW    0
0x0452      0x05F2      ANDWF    R2, 1
0x0453      0x05F3      ANDWF    R3, 1
0x0454      0x0870      MOVF     R0, 0
0x0455      0x00B5      MOVWF    __Lib_CStdlib_randx
0x0456      0x0871      MOVF     R1, 0
0x0457      0x00B6      MOVWF    __Lib_CStdlib_randx+1
0x0458      0x0872      MOVF     R2, 0
0x0459      0x00B7      MOVWF    __Lib_CStdlib_randx+2
0x045A      0x0873      MOVF     R3, 0
0x045B      0x00B8      MOVWF    __Lib_CStdlib_randx+3
;__Lib_CStdlib.c,311 ::
;__Lib_CStdlib.c,312 ::
L_end_rand:
0x045C      0x0008      RETURN
; end of _rand
_Int2Double:
;__Lib_MathDouble.c,1802 ::
;__Lib_MathDouble.c,1808 ::
0x045D      0x0020      MOVLB    0
;__Lib_MathDouble.c,1809 ::
0x045E      0x0870      MOVF     R0, 0
;__Lib_MathDouble.c,1810 ::
```

```
0x045F          0x00F8          MOVWF      R8
;__Lib_MathDouble.c,1811 ::
0x0460          0x0871          MOVF       R1, 0
;__Lib_MathDouble.c,1812 ::
0x0461          0x00F0          MOVWF      R0
;__Lib_MathDouble.c,1813 ::
0x0462          0x01F1          CLRF       R1
;__Lib_MathDouble.c,1814 ::
0x0463          0x01F2          CLRF       R2
;__Lib_MathDouble.c,1815 ::
0x0464          0x01F3          CLRF       R3
;__Lib_MathDouble.c,1818 ::
0x0465          0x01FB          CLRF       R11
;__Lib_MathDouble.c,1819 ::
0x0466          0x01FC          CLRF       R12
;__Lib_MathDouble.c,1821 ::
0x0467          0x3096          MOVLW      150
;__Lib_MathDouble.c,1822 ::
0x0468          0x00F3          MOVWF      R3
;__Lib_MathDouble.c,1823 ::
0x0469          0x01FA          CLRF       R10
;__Lib_MathDouble.c,1824 ::
0x046A          0x1FF0          BTFSS      R0, 7
;__Lib_MathDouble.c,1825 ::
0x046B          0x2C72          GOTO       FLO1632EEE
;__Lib_MathDouble.c,1827 ::
0x046C          0x09F8          COMF       R8, 1
;__Lib_MathDouble.c,1828 ::
0x046D          0x09F0          COMF       R0, 1
;__Lib_MathDouble.c,1830 ::
0x046E          0x0AF8          INCF       R8, 1
;__Lib_MathDouble.c,1831 ::
0x046F          0x1903          BTFSC      STATUS, 2
;__Lib_MathDouble.c,1832 ::
0x0470          0x0AF0          INCF       R0, 1
;__Lib_MathDouble.c,1834 ::
0x0471          0x17FA          BSF        R10, 7
;__Lib_MathDouble.c,1835 ::
FLO1632EEE:
;__Lib_MathDouble.c,1836 ::
0x0472          0x0870          MOVF       R0, 0
;__Lib_MathDouble.c,1837 ::
0x0473          0x00F1          MOVWF      R1
;__Lib_MathDouble.c,1838 ::
0x0474          0x0878          MOVF       R8, 0
;__Lib_MathDouble.c,1839 ::
0x0475          0x00F0          MOVWF      R0
```

```
;__Lib_MathDouble.c,1840 ::
0x0476	0x01F8	CLRF	R8
;__Lib_MathDouble.c,1841 ::
0x0477	0x01F2	CLRF	R2
;__Lib_MathDouble.c,1843 ::
0x0478	0x21CA	CALL	_NRM3232
;__Lib_MathDouble.c,1844 ::
L_end_Int2Double:
0x0479	0x0008	RETURN
; end of _Int2Double
_Add_32x32_FP:
;__Lib_MathDouble.c,379 ::
;__Lib_MathDouble.c,390 ::
;__Lib_MathDouble.c,392 ::
0x047A	0x0020	MOVLB	0
;__Lib_MathDouble.c,393 ::
0x047B	0x0872	MOVF	R2, 0
;__Lib_MathDouble.c,394 ::
0x047C	0x0676	XORWF	R6, 0
;__Lib_MathDouble.c,395 ::
0x047D	0x00A0	MOVWF	__math_tempbD
;__Lib_MathDouble.c,397 ::
0x047E	0x01F8	CLRF	R8
;__Lib_MathDouble.c,398 ::
0x047F	0x01F9	CLRF	R9
;__Lib_MathDouble.c,400 ::
0x0480	0x0873	MOVF	R3, 0
;__Lib_MathDouble.c,401 ::
0x0481	0x0277	SUBWF	R7, 0
;__Lib_MathDouble.c,402 ::
0x0482	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,403 ::
0x0483	0x2C9C	GOTO	USEA32
;__Lib_MathDouble.c,405 ::
0x0484	0x0877	MOVF	R7, 0
;__Lib_MathDouble.c,406 ::
0x0485	0x00FD	MOVWF	R13
;__Lib_MathDouble.c,407 ::
0x0486	0x0873	MOVF	R3, 0
;__Lib_MathDouble.c,408 ::
0x0487	0x00F7	MOVWF	R7
;__Lib_MathDouble.c,409 ::
0x0488	0x087D	MOVF	R13, 0
;__Lib_MathDouble.c,410 ::
0x0489	0x00F3	MOVWF	R3
;__Lib_MathDouble.c,412 ::
0x048A	0x0876	MOVF	R6, 0
```

```
;__Lib_MathDouble.c,413 ::
0x048B        0x00FD        MOVWF     R13
;__Lib_MathDouble.c,414 ::
0x048C        0x0872        MOVF      R2, 0
;__Lib_MathDouble.c,415 ::
0x048D        0x00F6        MOVWF     R6
;__Lib_MathDouble.c,416 ::
0x048E        0x087D        MOVF      R13, 0
;__Lib_MathDouble.c,417 ::
0x048F        0x00F2        MOVWF     R2
;__Lib_MathDouble.c,419 ::
0x0490        0x0875        MOVF      R5, 0
;__Lib_MathDouble.c,420 ::
0x0491        0x00FD        MOVWF     R13
;__Lib_MathDouble.c,421 ::
0x0492        0x0871        MOVF      R1, 0
;__Lib_MathDouble.c,422 ::
0x0493        0x00F5        MOVWF     R5
;__Lib_MathDouble.c,423 ::
0x0494        0x087D        MOVF      R13, 0
;__Lib_MathDouble.c,424 ::
0x0495        0x00F1        MOVWF     R1
;__Lib_MathDouble.c,426 ::
0x0496        0x0874        MOVF      R4, 0
;__Lib_MathDouble.c,427 ::
0x0497        0x00FD        MOVWF     R13
;__Lib_MathDouble.c,428 ::
0x0498        0x0870        MOVF      R0, 0
;__Lib_MathDouble.c,429 ::
0x0499        0x00F4        MOVWF     R4
;__Lib_MathDouble.c,430 ::
0x049A        0x087D        MOVF      R13, 0
;__Lib_MathDouble.c,431 ::
0x049B        0x00F0        MOVWF     R0
;__Lib_MathDouble.c,432 ::
USEA32:
;__Lib_MathDouble.c,433 ::
0x049C        0x0877        MOVF      R7, 0
;__Lib_MathDouble.c,434 ::
0x049D        0x1903        BTFSC     STATUS, 2
;__Lib_MathDouble.c,435 ::
0x049E        0x2D15        GOTO      JMPEEE
;__Lib_MathDouble.c,436 ::
0x049F        0x0872        MOVF      R2, 0
;__Lib_MathDouble.c,437 ::
0x04A0        0x00FA        MOVWF     R10
;__Lib_MathDouble.c,438 ::
```

```
0x04A1          0x17F2          BSF       R2, 7
;__Lib_MathDouble.c,439 ::
0x04A2          0x17F6          BSF       R6, 7
;__Lib_MathDouble.c,440 ::
0x04A3          0x0877          MOVF      R7, 0
;__Lib_MathDouble.c,441 ::
0x04A4          0x0273          SUBWF     R3, 0
;__Lib_MathDouble.c,442 ::
0x04A5          0x00F7          MOVWF     R7
;__Lib_MathDouble.c,443 ::
0x04A6          0x1903          BTFSC     STATUS, 2
;__Lib_MathDouble.c,444 ::
0x04A7          0x2CCF          GOTO      ALIGNED32
;__Lib_MathDouble.c,445 ::
0x04A8          0x3008          MOVLW     8
;__Lib_MathDouble.c,446 ::
0x04A9          0x0277          SUBWF     R7, 0
;__Lib_MathDouble.c,447 ::
0x04AA          0x1C03          BTFSS     STATUS, 0
;__Lib_MathDouble.c,448 ::
0x04AB          0x2CC5          GOTO      ALIGNB32
;__Lib_MathDouble.c,449 ::
0x04AC          0x00F7          MOVWF     R7
;__Lib_MathDouble.c,450 ::
0x04AD          0x0874          MOVF      R4, 0
;__Lib_MathDouble.c,451 ::
0x04AE          0x00F9          MOVWF     R9
;__Lib_MathDouble.c,452 ::
0x04AF          0x0875          MOVF      R5, 0
;__Lib_MathDouble.c,453 ::
0x04B0          0x00F4          MOVWF     R4
;__Lib_MathDouble.c,454 ::
0x04B1          0x0876          MOVF      R6, 0
;__Lib_MathDouble.c,455 ::
0x04B2          0x00F5          MOVWF     R5
;__Lib_MathDouble.c,456 ::
0x04B3          0x01F6          CLRF      R6
;__Lib_MathDouble.c,457 ::
0x04B4          0x3008          MOVLW     8
;__Lib_MathDouble.c,458 ::
0x04B5          0x0277          SUBWF     R7, 0
;__Lib_MathDouble.c,459 ::
0x04B6          0x1C03          BTFSS     STATUS, 0
;__Lib_MathDouble.c,460 ::
0x04B7          0x2CC5          GOTO      ALIGNB32
;__Lib_MathDouble.c,461 ::
0x04B8          0x00F7          MOVWF     R7
```

```
;__Lib_MathDouble.c,462 ::
0x04B9  0x0874  MOVF    R4, 0
;__Lib_MathDouble.c,463 ::
0x04BA  0x00F9  MOVWF   R9
;__Lib_MathDouble.c,464 ::
0x04BB  0x0875  MOVF    R5, 0
;__Lib_MathDouble.c,465 ::
0x04BC  0x00F4  MOVWF   R4
;__Lib_MathDouble.c,466 ::
0x04BD  0x01F5  CLRF    R5
;__Lib_MathDouble.c,467 ::
0x04BE  0x3008  MOVLW   8
;__Lib_MathDouble.c,468 ::
0x04BF  0x0277  SUBWF   R7, 0
;__Lib_MathDouble.c,469 ::
0x04C0  0x1C03  BTFSS   STATUS, 0
;__Lib_MathDouble.c,470 ::
0x04C1  0x2CC5  GOTO    ALIGNB32
;__Lib_MathDouble.c,471 ::
0x04C2  0x087A  MOVF    R10, 0
;__Lib_MathDouble.c,472 ::
0x04C3  0x00F2  MOVWF   R2
;__Lib_MathDouble.c,473 ::
0x04C4  0x2D15  GOTO    JMPEEE
;__Lib_MathDouble.c,474 ::
ALIGNB32:
;__Lib_MathDouble.c,475 ::
0x04C5  0x0877  MOVF    R7, 0
;__Lib_MathDouble.c,476 ::
0x04C6  0x1903  BTFSC   STATUS, 2
;__Lib_MathDouble.c,477 ::
0x04C7  0x2CCF  GOTO    ALIGNED32
;__Lib_MathDouble.c,478 ::
ALOOPB32:
;__Lib_MathDouble.c,479 ::
0x04C8  0x1003  BCF     STATUS, 0
;__Lib_MathDouble.c,480 ::
0x04C9  0x0CF6  RRF     R6, 1
;__Lib_MathDouble.c,481 ::
0x04CA  0x0CF5  RRF     R5, 1
;__Lib_MathDouble.c,482 ::
0x04CB  0x0CF4  RRF     R4, 1
;__Lib_MathDouble.c,483 ::
0x04CC  0x0CF9  RRF     R9, 1
;__Lib_MathDouble.c,484 ::
0x04CD  0x0BF7  DECFSZ  R7, 1
;__Lib_MathDouble.c,485 ::
```

```
0x04CE	0x2CC8	GOTO	ALOOPB32
;__Lib_MathDouble.c,486 ::
ALIGNED32:
;__Lib_MathDouble.c,487 ::
0x04CF	0x1FA0	BTFSS	__math_tempbD, 7
;__Lib_MathDouble.c,488 ::
0x04D0	0x2CDC	GOTO	AOK32
;__Lib_MathDouble.c,489 ::
0x04D1	0x09F9	COMF	R9, 1
;__Lib_MathDouble.c,490 ::
0x04D2	0x09F4	COMF	R4, 1
;__Lib_MathDouble.c,491 ::
0x04D3	0x09F5	COMF	R5, 1
;__Lib_MathDouble.c,492 ::
0x04D4	0x09F6	COMF	R6, 1
;__Lib_MathDouble.c,493 ::
0x04D5	0x0AF9	INCF	R9, 1
;__Lib_MathDouble.c,494 ::
0x04D6	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,495 ::
0x04D7	0x0AF4	INCF	R4, 1
;__Lib_MathDouble.c,496 ::
0x04D8	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,497 ::
0x04D9	0x0AF5	INCF	R5, 1
;__Lib_MathDouble.c,498 ::
0x04DA	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,499 ::
0x04DB	0x0AF6	INCF	R6, 1
;__Lib_MathDouble.c,500 ::
AOK32:
;__Lib_MathDouble.c,501 ::
0x04DC	0x0879	MOVF	R9, 0
;__Lib_MathDouble.c,502 ::
0x04DD	0x07F8	ADDWF	R8, 1
;__Lib_MathDouble.c,503 ::
0x04DE	0x0874	MOVF	R4, 0
;__Lib_MathDouble.c,504 ::
0x04DF	0x1803	BTFSC	STATUS, 0
;__Lib_MathDouble.c,505 ::
0x04E0	0x0F74	INCFSZ	R4, 0
;__Lib_MathDouble.c,506 ::
0x04E1	0x07F0	ADDWF	R0, 1
;__Lib_MathDouble.c,507 ::
0x04E2	0x0875	MOVF	R5, 0
;__Lib_MathDouble.c,508 ::
0x04E3	0x1803	BTFSC	STATUS, 0
```

```
;__Lib_MathDouble.c,509 ::
0x04E4	0x0F75	INCFSZ	R5, 0
;__Lib_MathDouble.c,510 ::
0x04E5	0x07F1	ADDWF	R1, 1
;__Lib_MathDouble.c,511 ::
0x04E6	0x0876	MOVF	R6, 0
;__Lib_MathDouble.c,512 ::
0x04E7	0x1803	BTFSC	STATUS, 0
;__Lib_MathDouble.c,513 ::
0x04E8	0x0F76	INCFSZ	R6, 0
;__Lib_MathDouble.c,514 ::
0x04E9	0x07F2	ADDWF	R2, 1
;__Lib_MathDouble.c,515 ::
0x04EA	0x1BA0	BTFSC	__math_tempbD, 7
;__Lib_MathDouble.c,516 ::
0x04EB	0x2CF5	GOTO	ACOMP32
;__Lib_MathDouble.c,517 ::
0x04EC	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,518 ::
0x04ED	0x2D05	GOTO	JMPNRMRND4032
;__Lib_MathDouble.c,519 ::
0x04EE	0x0CF2	RRF	R2, 1
;__Lib_MathDouble.c,520 ::
0x04EF	0x0CF1	RRF	R1, 1
;__Lib_MathDouble.c,521 ::
0x04F0	0x0CF0	RRF	R0, 1
;__Lib_MathDouble.c,522 ::
0x04F1	0x0CF8	RRF	R8, 1
;__Lib_MathDouble.c,523 ::
0x04F2	0x0FF3	INCFSZ	R3, 1
;__Lib_MathDouble.c,524 ::
0x04F3	0x2D05	GOTO	JMPNRMRND4032
;__Lib_MathDouble.c,525 ::
0x04F4	0x2D11	GOTO	ADD_JMPSETFOV32
;__Lib_MathDouble.c,526 ::
ACOMP32:
;__Lib_MathDouble.c,527 ::
0x04F5	0x1803	BTFSC	STATUS, 0
;__Lib_MathDouble.c,528 ::
0x04F6	0x2D09	GOTO	JMPNRM4032
;__Lib_MathDouble.c,529 ::
0x04F7	0x09F8	COMF	R8, 1
;__Lib_MathDouble.c,530 ::
0x04F8	0x09F0	COMF	R0, 1
;__Lib_MathDouble.c,531 ::
0x04F9	0x09F1	COMF	R1, 1
;__Lib_MathDouble.c,532 ::
```

```
0x04FA	0x09F2	COMF	R2, 1
;__Lib_MathDouble.c,533 ::
0x04FB	0x0AF8	INCF	R8, 1
;__Lib_MathDouble.c,534 ::
0x04FC	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,535 ::
0x04FD	0x0AF0	INCF	R0, 1
;__Lib_MathDouble.c,536 ::
0x04FE	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,537 ::
0x04FF	0x0AF1	INCF	R1, 1
;__Lib_MathDouble.c,538 ::
0x0500	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,539 ::
0x0501	0x0AF2	INCF	R2, 1
;__Lib_MathDouble.c,540 ::
0x0502	0x3080	MOVLW	128
;__Lib_MathDouble.c,541 ::
0x0503	0x06FA	XORWF	R10, 1
;__Lib_MathDouble.c,542 ::
0x0504	0x2D0D	GOTO	JMPNRM32
;__Lib_MathDouble.c,543 ::
JMPNRMRND4032:
;__Lib_MathDouble.c,544 ::
0x0505	0x3001	MOVLW	1
;__Lib_MathDouble.c,545 ::
0x0506	0x00FC	MOVWF	R12
;__Lib_MathDouble.c,547 ::
0x0507	0x2120	CALL	_NRM4032
;__Lib_MathDouble.c,549 ::
0x0508	0x2D17	GOTO	ADD32EEE
;__Lib_MathDouble.c,550 ::
JMPNRM4032:
;__Lib_MathDouble.c,551 ::
0x0509	0x3002	MOVLW	2
;__Lib_MathDouble.c,552 ::
0x050A	0x00FC	MOVWF	R12
;__Lib_MathDouble.c,554 ::
0x050B	0x2120	CALL	_NRM4032
;__Lib_MathDouble.c,556 ::
0x050C	0x2D17	GOTO	ADD32EEE
;__Lib_MathDouble.c,557 ::
JMPNRM32:
;__Lib_MathDouble.c,558 ::
0x050D	0x3004	MOVLW	4
;__Lib_MathDouble.c,559 ::
0x050E	0x00FC	MOVWF	R12
```

```
;__Lib_MathDouble.c,561 ::
0x050F          0x21CA          CALL       _NRM3232
;__Lib_MathDouble.c,563 ::
0x0510          0x2D17          GOTO       ADD32EEE
;__Lib_MathDouble.c,564 ::
ADD_JMPSETFOV32:
;__Lib_MathDouble.c,565 ::
0x0511          0x3008          MOVLW      8
;__Lib_MathDouble.c,566 ::
0x0512          0x00FC          MOVWF      R12
;__Lib_MathDouble.c,568 ::
0x0513          0x2017          CALL       _SETFOV32
;__Lib_MathDouble.c,570 ::
0x0514          0x2D17          GOTO       ADD32EEE
;__Lib_MathDouble.c,571 ::
JMPEEE:
;__Lib_MathDouble.c,572 ::
0x0515          0x3000          MOVLW      0
;__Lib_MathDouble.c,573 ::
0x0516          0x00FC          MOVWF      R12
;__Lib_MathDouble.c,574 ::
ADD32EEE:
;__Lib_MathDouble.c,575 ::
0x0517          0x0000          NOP
;__Lib_MathDouble.c,583 ::
L_end_Add_32x32_FP:
0x0518          0x0008          RETURN
; end of _Add_32x32_FP
_Word2Double:
;__Lib_MathDouble.c,1851 ::
;__Lib_MathDouble.c,1857 ::
0x0519          0x0020          MOVLB      0
;__Lib_MathDouble.c,1858 ::
0x051A          0x0870          MOVF       R0, 0
;__Lib_MathDouble.c,1859 ::
0x051B          0x00F8          MOVWF      R8
;__Lib_MathDouble.c,1860 ::
0x051C          0x0871          MOVF       R1, 0
;__Lib_MathDouble.c,1861 ::
0x051D          0x00F0          MOVWF      R0
;__Lib_MathDouble.c,1862 ::
0x051E          0x01F1          CLRF       R1
;__Lib_MathDouble.c,1863 ::
0x051F          0x01F2          CLRF       R2
;__Lib_MathDouble.c,1864 ::
0x0520          0x01F3          CLRF       R3
;__Lib_MathDouble.c,1867 ::
```

```
0x0521          0x01FB          CLRF      R11
;__Lib_MathDouble.c,1868 ::
0x0522          0x01FC          CLRF      R12
;__Lib_MathDouble.c,1870 ::
0x0523          0x3096          MOVLW     150
;__Lib_MathDouble.c,1871 ::
0x0524          0x00F3          MOVWF     R3
;__Lib_MathDouble.c,1872 ::
0x0525          0x01FA          CLRF      R10
;__Lib_MathDouble.c,1874 ::
0x0526          0x0870          MOVF      R0, 0
;__Lib_MathDouble.c,1875 ::
0x0527          0x00F1          MOVWF     R1
;__Lib_MathDouble.c,1876 ::
0x0528          0x0878          MOVF      R8, 0
;__Lib_MathDouble.c,1877 ::
0x0529          0x00F0          MOVWF     R0
;__Lib_MathDouble.c,1878 ::
0x052A          0x01F8          CLRF      R8
;__Lib_MathDouble.c,1879 ::
0x052B          0x01F2          CLRF      R2
;__Lib_MathDouble.c,1881 ::
0x052C          0x21CA          CALL      _NRM3232
;__Lib_MathDouble.c,1882 ::
L_end_Word2Double:
0x052D          0x0008          RETURN
; end of _Word2Double
_Div_32x32_FP:
;__Lib_MathDouble.c,758 ::
;__Lib_MathDouble.c,770 ::
;__Lib_MathDouble.c,773 ::
0x052E          0x0874          MOVF      R4, 0
0x052F          0x3A00          XORLW     0
0x0530          0x1D03          BTFSS     STATUS, 2
0x0531          0x2D47          GOTO      L_Div_32x32_FP7
;__Lib_MathDouble.c,774 ::
0x0532          0x0875          MOVF      R5, 0
0x0533          0x3A00          XORLW     0
0x0534          0x1D03          BTFSS     STATUS, 2
0x0535          0x2D47          GOTO      L_Div_32x32_FP8
;__Lib_MathDouble.c,775 ::
0x0536          0x0876          MOVF      R6, 0
0x0537          0x3A00          XORLW     0
0x0538          0x1D03          BTFSS     STATUS, 2
0x0539          0x2D47          GOTO      L_Div_32x32_FP9
;__Lib_MathDouble.c,776 ::
0x053A          0x0877          MOVF      R7, 0
```

```
0x053B	0x3A00	XORLW	0
0x053C	0x1D03	BTFSS	STATUS, 2
0x053D	0x2D47	GOTO	L_Div_32x32_FP10
;__Lib_MathDouble.c,777 ::
0x053E	0x30FF	MOVLW	255
0x053F	0x00F0	MOVWF	R0
;__Lib_MathDouble.c,778 ::
0x0540	0x30FF	MOVLW	255
0x0541	0x00F1	MOVWF	R1
;__Lib_MathDouble.c,779 ::
0x0542	0x307F	MOVLW	127
0x0543	0x00F2	MOVWF	R2
;__Lib_MathDouble.c,780 ::
0x0544	0x30FF	MOVLW	255
0x0545	0x00F3	MOVWF	R3
;__Lib_MathDouble.c,781 ::
0x0546	0x2DE1	GOTO	L_end_Div_32x32_FP
;__Lib_MathDouble.c,782 ::
L_Div_32x32_FP10:
L_Div_32x32_FP9:
L_Div_32x32_FP8:
L_Div_32x32_FP7:
;__Lib_MathDouble.c,785 ::
0x0547	0x0020	MOVLB	0
;__Lib_MathDouble.c,787 ::
0x0548	0x0877	MOVF	R7, 0
;__Lib_MathDouble.c,788 ::
0x0549	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,789 ::
0x054A	0x2DD5	GOTO	JPDSETFDZ32
;__Lib_MathDouble.c,791 ::
0x054B	0x0873	MOVF	R3, 0
;__Lib_MathDouble.c,792 ::
0x054C	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,793 ::
0x054D	0x2DD8	GOTO	JPDRES032
;__Lib_MathDouble.c,795 ::
0x054E	0x0872	MOVF	R2, 0
;__Lib_MathDouble.c,796 ::
0x054F	0x0676	XORWF	R6, 0
;__Lib_MathDouble.c,797 ::
0x0550	0x3980	ANDLW	128
;__Lib_MathDouble.c,798 ::
0x0551	0x00FA	MOVWF	R10
;__Lib_MathDouble.c,799 ::
0x0552	0x17F2	BSF	R2, 7
;__Lib_MathDouble.c,800 ::
```

```
0x0553          0x17F6          BSF       R6, 7
;__Lib_MathDouble.c,802 ::
0x0554          0x01A0          CLRF      __math_tempbD
;__Lib_MathDouble.c,803 ::
0x0555          0x0872          MOVF      R2, 0
;__Lib_MathDouble.c,804 ::
0x0556          0x00F8          MOVWF     R8
;__Lib_MathDouble.c,805 ::
0x0557          0x0871          MOVF      R1, 0
;__Lib_MathDouble.c,806 ::
0x0558          0x00FC          MOVWF     R12
;__Lib_MathDouble.c,807 ::
0x0559          0x0870          MOVF      R0, 0
;__Lib_MathDouble.c,808 ::
0x055A          0x00FD          MOVWF     R13
;__Lib_MathDouble.c,810 ::
0x055B          0x0874          MOVF      R4, 0
;__Lib_MathDouble.c,811 ::
0x055C          0x02FD          SUBWF     R13, 1
;__Lib_MathDouble.c,812 ::
0x055D          0x0875          MOVF      R5, 0
;__Lib_MathDouble.c,813 ::
0x055E          0x1C03          BTFSS     STATUS, 0
;__Lib_MathDouble.c,814 ::
0x055F          0x0F75          INCFSZ    R5, 0
;__Lib_MathDouble.c,816 ::
0x0560          0x02FC          SUBWF     R12, 1
;__Lib_MathDouble.c,817 ::
0x0561          0x0876          MOVF      R6, 0
;__Lib_MathDouble.c,818 ::
0x0562          0x1C03          BTFSS     STATUS, 0
;__Lib_MathDouble.c,819 ::
0x0563          0x0F76          INCFSZ    R6, 0
;__Lib_MathDouble.c,821 ::
0x0564          0x02F8          SUBWF     R8, 1
;__Lib_MathDouble.c,822 ::
0x0565          0x01F8          CLRF      R8
;__Lib_MathDouble.c,823 ::
0x0566          0x01FC          CLRF      R12
;__Lib_MathDouble.c,824 ::
0x0567          0x01FD          CLRF      R13
;__Lib_MathDouble.c,825 ::
0x0568          0x1C03          BTFSS     STATUS, 0
;__Lib_MathDouble.c,826 ::
0x0569          0x2D71          GOTO      DALIGN32OK
;__Lib_MathDouble.c,828 ::
0x056A          0x1003          BCF       STATUS, 0
```

```
;__Lib_MathDouble.c,829 ::
0x056B	0x0CF2	RRF	R2, 1
;__Lib_MathDouble.c,830 ::
0x056C	0x0CF1	RRF	R1, 1
;__Lib_MathDouble.c,831 ::
0x056D	0x0CF0	RRF	R0, 1
;__Lib_MathDouble.c,832 ::
0x056E	0x0CF8	RRF	R8, 1
;__Lib_MathDouble.c,833 ::
0x056F	0x3001	MOVLW	1
;__Lib_MathDouble.c,834 ::
0x0570	0x00A0	MOVWF	__math_tempbD
;__Lib_MathDouble.c,835 ::
DALIGN32OK:
;__Lib_MathDouble.c,836 ::
0x0571	0x0877	MOVF	R7, 0
;__Lib_MathDouble.c,837 ::
0x0572	0x02F3	SUBWF	R3, 1
;__Lib_MathDouble.c,838 ::
0x0573	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,839 ::
0x0574	0x2D7B	GOTO	ALTB32
;__Lib_MathDouble.c,841 ::
0x0575	0x307E	MOVLW	126
;__Lib_MathDouble.c,842 ::
0x0576	0x0720	ADDWF	__math_tempbD, 0
;__Lib_MathDouble.c,843 ::
0x0577	0x07F3	ADDWF	R3, 1
;__Lib_MathDouble.c,844 ::
0x0578	0x1803	BTFSC	STATUS, 0
;__Lib_MathDouble.c,845 ::
0x0579	0x2DDB	GOTO	JPDSETFOV32
;__Lib_MathDouble.c,846 ::
0x057A	0x2D80	GOTO	DARGOK32
;__Lib_MathDouble.c,847 ::
ALTB32:
;__Lib_MathDouble.c,848 ::
0x057B	0x307E	MOVLW	126
;__Lib_MathDouble.c,849 ::
0x057C	0x0720	ADDWF	__math_tempbD, 0
;__Lib_MathDouble.c,850 ::
0x057D	0x07F3	ADDWF	R3, 1
;__Lib_MathDouble.c,851 ::
0x057E	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,852 ::
0x057F	0x2DDE	GOTO	JPDSETFUN32
;__Lib_MathDouble.c,853 ::
```

```
DARGOK32:
;__Lib_MathDouble.c,854 ::
0x0580          0x3018          MOVLW      24
;__Lib_MathDouble.c,856 ::
0x0581          0x07FA          ADDWF      R10, 1
;__Lib_MathDouble.c,857 ::
DLOOP32:
;__Lib_MathDouble.c,858 ::
0x0582          0x0DFD          RLF        R13, 1
;__Lib_MathDouble.c,859 ::
0x0583          0x0DFC          RLF        R12, 1
;__Lib_MathDouble.c,860 ::
0x0584          0x0DF8          RLF        R8, 1
;__Lib_MathDouble.c,861 ::
0x0585          0x0DF0          RLF        R0, 1
;__Lib_MathDouble.c,862 ::
0x0586          0x0DF1          RLF        R1, 1
;__Lib_MathDouble.c,863 ::
0x0587          0x0DF2          RLF        R2, 1
;__Lib_MathDouble.c,864 ::
0x0588          0x0DA0          RLF        __math_tempbD, 1
;__Lib_MathDouble.c,866 ::
0x0589          0x0874          MOVF       R4, 0
;__Lib_MathDouble.c,867 ::
0x058A          0x02F0          SUBWF      R0, 1
;__Lib_MathDouble.c,868 ::
0x058B          0x0875          MOVF       R5, 0
;__Lib_MathDouble.c,869 ::
0x058C          0x1C03          BTFSS      STATUS, 0
;__Lib_MathDouble.c,870 ::
0x058D          0x0F75          INCFSZ     R5, 0
;__Lib_MathDouble.c,872 ::
0x058E          0x02F1          SUBWF      R1, 1
;__Lib_MathDouble.c,873 ::
0x058F          0x0876          MOVF       R6, 0
;__Lib_MathDouble.c,874 ::
0x0590          0x1C03          BTFSS      STATUS, 0
;__Lib_MathDouble.c,875 ::
0x0591          0x0F76          INCFSZ     R6, 0
;__Lib_MathDouble.c,877 ::
0x0592          0x02F2          SUBWF      R2, 1
;__Lib_MathDouble.c,878 ::
0x0593          0x0D76          RLF        R6, 0
;__Lib_MathDouble.c,879 ::
0x0594          0x04A0          IORWF      __math_tempbD, 1
;__Lib_MathDouble.c,880 ::
0x0595          0x1C20          BTFSS      __math_tempbD, 0
```

```
;__Lib_MathDouble.c,881 ::
0x0596          0x2D99          GOTO       DREST32
;__Lib_MathDouble.c,882 ::
0x0597          0x147D          BSF        R13, 0
;__Lib_MathDouble.c,883 ::
0x0598          0x2DA4          GOTO       DOK32
;__Lib_MathDouble.c,884 ::
DREST32:
;__Lib_MathDouble.c,885 ::
0x0599          0x0874          MOVF       R4, 0
;__Lib_MathDouble.c,886 ::
0x059A          0x07F0          ADDWF      R0, 1
;__Lib_MathDouble.c,887 ::
0x059B          0x0875          MOVF       R5, 0
;__Lib_MathDouble.c,888 ::
0x059C          0x1803          BTFSC      STATUS, 0
;__Lib_MathDouble.c,889 ::
0x059D          0x0F75          INCFSZ     R5, 0
;__Lib_MathDouble.c,891 ::
0x059E          0x07F1          ADDWF      R1, 1
;__Lib_MathDouble.c,892 ::
0x059F          0x0876          MOVF       R6, 0
;__Lib_MathDouble.c,893 ::
0x05A0          0x1803          BTFSC      STATUS, 0
;__Lib_MathDouble.c,894 ::
0x05A1          0x0A76          INCF       R6, 0
;__Lib_MathDouble.c,895 ::
0x05A2          0x07F2          ADDWF      R2, 1
;__Lib_MathDouble.c,896 ::
0x05A3          0x107D          BCF        R13, 0
;__Lib_MathDouble.c,897 ::
DOK32:
;__Lib_MathDouble.c,899 ::
0x05A4          0x03FA          DECF       R10, 1
;__Lib_MathDouble.c,900 ::
0x05A5          0x301F          MOVLW      31
;__Lib_MathDouble.c,901 ::
0x05A6          0x057A          ANDWF      R10, 0
;__Lib_MathDouble.c,902 ::
0x05A7          0x1D03          BTFSS      STATUS, 2
;__Lib_MathDouble.c,903 ::
0x05A8          0x2D82          GOTO       DLOOP32
;__Lib_MathDouble.c,905 ::
0x05A9          0x1B7B          BTFSC      R11, 6
;__Lib_MathDouble.c,906 ::
0x05AA          0x1C7D          BTFSS      R13, 0
;__Lib_MathDouble.c,907 ::
```

```
0x05AB	0x2DCB	GOTO	DIV32OK
;__Lib_MathDouble.c,908 ::
0x05AC	0x1003	BCF	STATUS, 0
;__Lib_MathDouble.c,909 ::
0x05AD	0x0DF0	RLF	R0, 1
;__Lib_MathDouble.c,910 ::
0x05AE	0x0DF1	RLF	R1, 1
;__Lib_MathDouble.c,911 ::
0x05AF	0x0DF2	RLF	R2, 1
;__Lib_MathDouble.c,912 ::
0x05B0	0x0DA0	RLF	__math_tempbD, 1
;__Lib_MathDouble.c,914 ::
0x05B1	0x0874	MOVF	R4, 0
;__Lib_MathDouble.c,915 ::
0x05B2	0x02F0	SUBWF	R0, 1
;__Lib_MathDouble.c,916 ::
0x05B3	0x0875	MOVF	R5, 0
;__Lib_MathDouble.c,917 ::
0x05B4	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,918 ::
0x05B5	0x0F75	INCFSZ	R5, 0
;__Lib_MathDouble.c,919 ::
0x05B6	0x02F1	SUBWF	R1, 1
;__Lib_MathDouble.c,920 ::
0x05B7	0x0876	MOVF	R6, 0
;__Lib_MathDouble.c,921 ::
0x05B8	0x1C03	BTFSS	STATUS, 0
;__Lib_MathDouble.c,922 ::
0x05B9	0x0F76	INCFSZ	R6, 0
;__Lib_MathDouble.c,923 ::
0x05BA	0x02F2	SUBWF	R2, 1
;__Lib_MathDouble.c,924 ::
0x05BB	0x0D76	RLF	R6, 0
;__Lib_MathDouble.c,925 ::
0x05BC	0x0420	IORWF	__math_tempbD, 0
;__Lib_MathDouble.c,926 ::
0x05BD	0x3901	ANDLW	1
;__Lib_MathDouble.c,927 ::
0x05BE	0x07FD	ADDWF	R13, 1
;__Lib_MathDouble.c,928 ::
0x05BF	0x1803	BTFSC	STATUS, 0
;__Lib_MathDouble.c,929 ::
0x05C0	0x0AFC	INCF	R12, 1
;__Lib_MathDouble.c,930 ::
0x05C1	0x1903	BTFSC	STATUS, 2
;__Lib_MathDouble.c,931 ::
0x05C2	0x0AF8	INCF	R8, 1
```

```
;__Lib_MathDouble.c,932 ::
0x05C3        0x1D03        BTFSS     STATUS, 2
;__Lib_MathDouble.c,933 ::
0x05C4        0x2DCB        GOTO      DIV32OK
;__Lib_MathDouble.c,934 ::
0x05C5        0x0CF8        RRF       R8, 1
;__Lib_MathDouble.c,935 ::
0x05C6        0x0CFC        RRF       R12, 1
;__Lib_MathDouble.c,936 ::
0x05C7        0x0CFD        RRF       R13, 1
;__Lib_MathDouble.c,937 ::
0x05C8        0x0AF3        INCF      R3, 1
;__Lib_MathDouble.c,938 ::
0x05C9        0x1903        BTFSC     STATUS, 2
;__Lib_MathDouble.c,939 ::
0x05CA        0x2DDB        GOTO      JPDSETFOV32
;__Lib_MathDouble.c,940 ::
DIV32OK:
;__Lib_MathDouble.c,941 ::
0x05CB        0x1FFA        BTFSS     R10, 7
;__Lib_MathDouble.c,942 ::
0x05CC        0x13F8        BCF       R8, 7
;__Lib_MathDouble.c,943 ::
0x05CD        0x0878        MOVF      R8, 0
;__Lib_MathDouble.c,944 ::
0x05CE        0x00F2        MOVWF     R2
;__Lib_MathDouble.c,945 ::
0x05CF        0x087C        MOVF      R12, 0
;__Lib_MathDouble.c,946 ::
0x05D0        0x00F1        MOVWF     R1
;__Lib_MathDouble.c,947 ::
0x05D1        0x087D        MOVF      R13, 0
;__Lib_MathDouble.c,948 ::
0x05D2        0x00F0        MOVWF     R0
;__Lib_MathDouble.c,950 ::
0x05D3        0x3000        MOVLW     0
;__Lib_MathDouble.c,952 ::
0x05D4        0x2DE0        GOTO      DIV32EEE
;__Lib_MathDouble.c,953 ::
JPDSETFDZ32:
;__Lib_MathDouble.c,954 ::
0x05D5        0x3001        MOVLW     1
;__Lib_MathDouble.c,957 ::
0x05D6        0x20BD        CALL      _SETFDZ32
;__Lib_MathDouble.c,959 ::
0x05D7        0x2DE0        GOTO      DIV32EEE
;__Lib_MathDouble.c,960 ::
```

```
JPDRES032:
;__Lib_MathDouble.c,961 ::
0x05D8        0x3002        MOVLW     2
;__Lib_MathDouble.c,964 ::
0x05D9        0x2023        CALL      _RES032
;__Lib_MathDouble.c,966 ::
0x05DA        0x2DE0        GOTO      DIV32EEE
;__Lib_MathDouble.c,967 ::
JPDSETFOV32:
;__Lib_MathDouble.c,968 ::
0x05DB        0x3004        MOVLW     4
;__Lib_MathDouble.c,971 ::
0x05DC        0x2017        CALL      _SETFOV32
;__Lib_MathDouble.c,973 ::
0x05DD        0x2DE0        GOTO      DIV32EEE
;__Lib_MathDouble.c,974 ::
JPDSETFUN32:
;__Lib_MathDouble.c,975 ::
0x05DE        0x3008        MOVLW     8
;__Lib_MathDouble.c,978 ::
0x05DF        0x202A        CALL      _SETFUN32
;__Lib_MathDouble.c,980 ::
DIV32EEE:
;__Lib_MathDouble.c,981 ::
0x05E0        0x0000        NOP
;__Lib_MathDouble.c,990 ::
L_end_Div_32x32_FP:
0x05E1        0x0008        RETURN
; end of _Div_32x32_FP
_UART1_Write_Text:
;__Lib_UART_b21.c,61 ::
;__Lib_UART_b21.c,62 ::
0x05E2        0x0020        MOVLB     0
0x05E3        0x01CF        CLRF      UART1_Write_Text_counter_L0
;__Lib_UART_b21.c,64 ::
0x05E4        0x084C        MOVF      FARG_UART1_Write_Text_uart_text, 0
0x05E5        0x0084        MOVWF     FSR0
0x05E6        0x084D        MOVF      FARG_UART1_Write_Text_uart_text+1, 0
0x05E7        0x0085        MOVWF     FSR0H
0x05E8        0x0800        MOVF      INDF0, 0
0x05E9        0x00CE        MOVWF     UART1_Write_Text_data__L0
;__Lib_UART_b21.c,65 ::
L_UART1_Write_Text5:
0x05EA        0x084E        MOVF      UART1_Write_Text_data__L0, 0
0x05EB        0x3A00        XORLW     0
0x05EC        0x1903        BTFSC     STATUS, 2
0x05ED        0x2DFC        GOTO      L_UART1_Write_Text6
```

```
;__Lib_UART_b21.c,66 ::
0x05EE        0x084E        MOVF      UART1_Write_Text_data__L0, 0
0x05EF        0x00DA        MOVWF     FARG_UART1_Write_data_
0x05F0        0x20B3        CALL      _UART1_Write
;__Lib_UART_b21.c,67 ::
0x05F1        0x0020        MOVLB     0
0x05F2        0x0ACF        INCF      UART1_Write_Text_counter_L0, 1
;__Lib_UART_b21.c,68 ::
0x05F3        0x084F        MOVF      UART1_Write_Text_counter_L0, 0
0x05F4        0x074C        ADDWF     FARG_UART1_Write_Text_uart_text, 0
0x05F5        0x0084        MOVWF     FSR0
0x05F6        0x3000        MOVLW     0
0x05F7        0x3D4D        ADDWFC    FARG_UART1_Write_Text_uart_text+1, 0
0x05F8        0x0085        MOVWF     FSR0H
0x05F9        0x0800        MOVF      INDF0, 0
0x05FA        0x00CE        MOVWF     UART1_Write_Text_data__L0
;__Lib_UART_b21.c,69 ::
0x05FB        0x2DEA        GOTO      L_UART1_Write_Text5
L_UART1_Write_Text6:
;__Lib_UART_b21.c,70 ::
L_end_UART1_Write_Text:
0x05FC        0x0008        RETURN
; end of _UART1_Write_Text
___CC2DW:
;__Lib_System.c,71 ::
;__Lib_System.c,73 ::
_CC2DL_Loop1:
;__Lib_System.c,74 ::
0x05FD        0x0012        MOVIW     INDF0++
;__Lib_System.c,75 ::
0x05FE        0x001E        MOVWI     INDF1++
;__Lib_System.c,76 ::
0x05FF        0x03F0        DECF      R0, 1
;__Lib_System.c,77 ::
0x0600        0x1D03        BTFSS     STATUS, 2
;__Lib_System.c,78 ::
0x0601        0x2DFD        GOTO      _CC2DL_Loop1
;__Lib_System.c,79 ::
0x0602        0x03F1        DECF      R1, 1
;__Lib_System.c,80 ::
0x0603        0x1D03        BTFSS     STATUS, 2
;__Lib_System.c,81 ::
0x0604        0x2DFD        GOTO      _CC2DL_Loop1
;__Lib_System.c,83 ::
L_end___CC2DW:
0x0605        0x0008        RETURN
; end of ___CC2DW
```

```
_Enviar:
;PuenteH.c,91 ::          void Enviar(void)
;PuenteH.c,94 ::          FloatToStr(Respot,txt);
0x0606  0x0020    MOVLB   0
0x0607  0x082C    MOVF    _Respot, 0
0x0608  0x00CC    MOVWF   FARG_FloatToStr_fnum
0x0609  0x082D    MOVF    _Respot+1, 0
0x060A  0x00CD    MOVWF   FARG_FloatToStr_fnum+1
0x060B  0x082E    MOVF    _Respot+2, 0
0x060C  0x00CE    MOVWF   FARG_FloatToStr_fnum+2
0x060D  0x082F    MOVF    _Respot+3, 0
0x060E  0x00CF    MOVWF   FARG_FloatToStr_fnum+3
0x060F  0x3039    MOVLW   _txt
0x0610  0x00D0    MOVWF   FARG_FloatToStr_str
0x0611  0x3000    MOVLW   hi_addr(_txt)
0x0612  0x00D1    MOVWF   FARG_FloatToStr_str+1
0x0613  0x21F9    CALL    _FloatToStr
;PuenteH.c,95 ::          UART1_Write_Text(txt);
0x0614  0x3039    MOVLW   _txt
0x0615  0x00CC    MOVWF   FARG_UART1_Write_Text_uart_text
0x0616  0x3000    MOVLW   hi_addr(_txt)
0x0617  0x00CD    MOVWF   FARG_UART1_Write_Text_uart_text+1
0x0618  0x25E2    CALL    _UART1_Write_Text
;PuenteH.c,96 ::          UART1_Write(0x0D);UART1_Write(0x0A);
0x0619  0x300D    MOVLW   13
0x061A  0x00DA    MOVWF   FARG_UART1_Write_data_
0x061B  0x20B3    CALL    _UART1_Write
0x061C  0x300A    MOVLW   10
0x061D  0x0020    MOVLB   0
0x061E  0x00DA    MOVWF   FARG_UART1_Write_data_
0x061F  0x20B3    CALL    _UART1_Write
;PuenteH.c,97 ::          }
L_end_Enviar:
0x0620  0x0008    RETURN
; end of _Enviar
_ADC:
;PuenteH.c,65 ::          void ADC(void)
;PuenteH.c,68 ::          movlw 0x18 //24
0x0621  0x3018    MOVLW   24
;PuenteH.c,69 ::          movwf _aux;
0x0622  0x00B4    MOVWF   _aux
;PuenteH.c,70 ::          decfsz _aux,1
0x0623  0x0BB4    DECFSZ  _aux, 1
;PuenteH.c,71 ::          goto $-1
0x0624  0x2E23    GOTO    $-1
;PuenteH.c,72 ::          movlb 0x01 //Banco 1
0x0625  0x0021    MOVLB   1
```

```
;PuenteH.c,73 ::          bsf 0x9D,1 //GO/DONE
0x0626      0x149D       BSF       ADCON0, 1
;PuenteH.c,74 ::          btfsc 0x9D,1 //GO/DONE = 0????
0x0627      0x189D       BTFSC     ADCON0, 1
;PuenteH.c,75 ::          goto $-1
0x0628      0x2E27       GOTO      $-1
;PuenteH.c,77 ::          pot = ADRESH;
0x0629      0x0021       MOVLB     1
0x062A      0x081C       MOVF      ADRESH, 0
0x062B      0x0020       MOVLB     0
0x062C      0x00B0       MOVWF     _pot
0x062D      0x01B1       CLRF      _pot+1
;PuenteH.c,78 ::          pot = pot << 8;
0x062E      0x0830       MOVF      _pot, 0
0x062F      0x00F1       MOVWF     R1
0x0630      0x01F0       CLRF      R0
0x0631      0x0870       MOVF      R0, 0
0x0632      0x00B0       MOVWF     _pot
0x0633      0x0871       MOVF      R1, 0
0x0634      0x00B1       MOVWF     _pot+1
;PuenteH.c,79 ::          pot = pot + ADRESL;
0x0635      0x0021       MOVLB     1
0x0636      0x081B       MOVF      ADRESL, 0
0x0637      0x07F0       ADDWF     R0, 1
0x0638      0x3000       MOVLW     0
0x0639      0x3DF1       ADDWFC    R1, 1
0x063A      0x0870       MOVF      R0, 0
0x063B      0x0020       MOVLB     0
0x063C      0x00B0       MOVWF     _pot
0x063D      0x0871       MOVF      R1, 0
0x063E      0x00B1       MOVWF     _pot+1
;PuenteH.c,81 ::          Respot = (4.94)*pot;
0x063F      0x2519       CALL      _Word2Double
0x0640      0x307B       MOVLW     123
0x0641      0x00F4       MOVWF     R4
0x0642      0x3014       MOVLW     20
0x0643      0x00F5       MOVWF     R5
0x0644      0x301E       MOVLW     30
0x0645      0x00F6       MOVWF     R6
0x0646      0x3081       MOVLW     129
0x0647      0x00F7       MOVWF     R7
0x0648      0x20C0       CALL      _Mul_32x32_FP
0x0649      0x0870       MOVF      R0, 0
0x064A      0x00AC       MOVWF     _Respot
0x064B      0x0871       MOVF      R1, 0
0x064C      0x00AD       MOVWF     _Respot+1
0x064D      0x0872       MOVF      R2, 0
```

```
0x064E          0x00AE          MOVWF     _Respot+2
0x064F          0x0873          MOVF      R3, 0
0x0650          0x00AF          MOVWF     _Respot+3
;PuenteH.c,82 ::                Respot= Respot/1023;
0x0651          0x3000          MOVLW     0
0x0652          0x00F4          MOVWF     R4
0x0653          0x30C0          MOVLW     192
0x0654          0x00F5          MOVWF     R5
0x0655          0x307F          MOVLW     127
0x0656          0x00F6          MOVWF     R6
0x0657          0x3088          MOVLW     136
0x0658          0x00F7          MOVWF     R7
0x0659          0x252E          CALL      _Div_32x32_FP
0x065A          0x0870          MOVF      R0, 0
0x065B          0x00AC          MOVWF     _Respot
0x065C          0x0871          MOVF      R1, 0
0x065D          0x00AD          MOVWF     _Respot+1
0x065E          0x0872          MOVF      R2, 0
0x065F          0x00AE          MOVWF     _Respot+2
0x0660          0x0873          MOVF      R3, 0
0x0661          0x00AF          MOVWF     _Respot+3
;PuenteH.c,85 ::                ruido = rand();
0x0662          0x241F          CALL      _rand
0x0663          0x245D          CALL      _Int2Double
0x0664          0x0870          MOVF      R0, 0
0x0665          0x00C8          MOVWF     _ruido
0x0666          0x0871          MOVF      R1, 0
0x0667          0x00C9          MOVWF     _ruido+1
0x0668          0x0872          MOVF      R2, 0
0x0669          0x00CA          MOVWF     _ruido+2
0x066A          0x0873          MOVF      R3, 0
0x066B          0x00CB          MOVWF     _ruido+3
;PuenteH.c,86 ::                ruido = ruido * 0.000001;
0x066C          0x30BD          MOVLW     189
0x066D          0x00F4          MOVWF     R4
0x066E          0x3037          MOVLW     55
0x066F          0x00F5          MOVWF     R5
0x0670          0x3006          MOVLW     6
0x0671          0x00F6          MOVWF     R6
0x0672          0x306B          MOVLW     107
0x0673          0x00F7          MOVWF     R7
0x0674          0x20C0          CALL      _Mul_32x32_FP
0x0675          0x0870          MOVF      R0, 0
0x0676          0x00C8          MOVWF     _ruido
0x0677          0x0871          MOVF      R1, 0
0x0678          0x00C9          MOVWF     _ruido+1
0x0679          0x0872          MOVF      R2, 0
```

```
0x067A      0x00CA      MOVWF    _ruido+2
0x067B      0x0873      MOVF     R3, 0
0x067C      0x00CB      MOVWF    _ruido+3
;PuenteH.c,87 ::        Respot = Respot + ruido;
0x067D      0x082C      MOVF     _Respot, 0
0x067E      0x00F4      MOVWF    R4
0x067F      0x082D      MOVF     _Respot+1, 0
0x0680      0x00F5      MOVWF    R5
0x0681      0x082E      MOVF     _Respot+2, 0
0x0682      0x00F6      MOVWF    R6
0x0683      0x082F      MOVF     _Respot+3, 0
0x0684      0x00F7      MOVWF    R7
0x0685      0x247A      CALL     _Add_32x32_FP
0x0686      0x0870      MOVF     R0, 0
0x0687      0x00AC      MOVWF    _Respot
0x0688      0x0871      MOVF     R1, 0
0x0689      0x00AD      MOVWF    _Respot+1
0x068A      0x0872      MOVF     R2, 0
0x068B      0x00AE      MOVWF    _Respot+2
0x068C      0x0873      MOVF     R3, 0
0x068D      0x00AF      MOVWF    _Respot+3
;PuenteH.c,89 ::        }
L_end_ADC:
0x068E      0x0008      RETURN
; end of _ADC
_Configuracion:
;PuenteH.c,16 ::        void Configuracion(void)
;PuenteH.c,20 ::        movlb 0x01 //Banco 1
0x068F      0x0021      MOVLB    1
;PuenteH.c,21 ::        movlw 0x73
0x0690      0x3073      MOVLW    115
;PuenteH.c,22 ::        movwf 0x99
0x0691      0x0099      MOVWF    OSCCON
;PuenteH.c,23 ::        btfss 0x9A,4
0x0692      0x1E1A      BTFSS    OSCSTAT, 4
;PuenteH.c,24 ::        goto $-1
0x0693      0x2E92      GOTO     $-1
;PuenteH.c,25 ::        btfss 0x9A,3
0x0694      0x1D9A      BTFSS    OSCSTAT, 3
;PuenteH.c,26 ::        goto $-1
0x0695      0x2E94      GOTO     $-1
;PuenteH.c,27 ::        btfss 0x9A,0
0x0696      0x1C1A      BTFSS    OSCSTAT, 0
;PuenteH.c,28 ::        goto $-1
0x0697      0x2E96      GOTO     $-1
;PuenteH.c,30 ::        movlw 0x08 //RA3 -> Entrada
0x0698      0x3008      MOVLW    8
```

```
;PuenteH.c,31 ::          movwf 0x8C //TRISA
0x0699      0x008C       MOVWF     TRISA
;PuenteH.c,32 ::          movlw 0x02//
0x069A      0x3002       MOVLW     2
;PuenteH.c,33 ::          movwf 0x8D //TRISB
0x069B      0x008D       MOVWF     TRISB
;PuenteH.c,35 ::          movlw 0x0D //AN3
0x069C      0x300D       MOVLW     13
;PuenteH.c,36 ::          movwf 0x9D //ADCON0
0x069D      0x009D       MOVWF     ADCON0
;PuenteH.c,37 ::          movlw 0x90 //fosc/8 (1 microseg),Right
0x069E      0x3090       MOVLW     144
;PuenteH.c,38 ::          movwf 0x9E //ADCON1
0x069F      0x009E       MOVWF     ADCON1
;PuenteH.c,40 ::          movlw 0x85
0x06A0      0x3085       MOVLW     133
;PuenteH.c,41 ::          movwf 0x95 //OPTION_REG (prescalador = 64)
0x06A1      0x0095       MOVWF     OPTION_REG
;PuenteH.c,43 ::          movlb 0x03 //Banco 3
0x06A2      0x0023       MOVLB     3
;PuenteH.c,44 ::          movlw 0x08 //RA3 -> Analogico
0x06A3      0x3008       MOVLW     8
;PuenteH.c,45 ::          movwf 0x18C //ANSELA
0x06A4      0x008C       MOVWF     ANSELA
;PuenteH.c,46 ::          clrf 0x18D //ANSELB
0x06A5      0x018D       CLRF      ANSELB
;PuenteH.c,48 ::          movlw 0x19 //38400 (0x0C),19200 (0x19), 9600 (0x0C)
0x06A6      0x3019       MOVLW     25
;PuenteH.c,49 ::          movwf 0x19B //SPBRGL
0x06A7      0x009B       MOVWF     SPBRG
;PuenteH.c,50 ::          bsf 0x19D,7 //RCSTA (SPEN=1)
0x06A8      0x179D       BSF       RCSTA, 7
;PuenteH.c,51 ::          bsf 0x19D,4 //RCSTA (CREN=1)
0x06A9      0x161D       BSF       RCSTA, 4
;PuenteH.c,52 ::          bsf 0x19E,5 //TXSTA (TXEN=1)
0x06AA      0x169E       BSF       TXSTA, 5
;PuenteH.c,53 ::          bsf 0x19E,2 //TXSTA (BRGH=1) <=> 38400 y 19200
0x06AB      0x151E       BSF       TXSTA, 2
;PuenteH.c,54 ::          movlb 0x00 //Banco 0
0x06AC      0x0020       MOVLB     0
;PuenteH.c,56 ::          movlw 0xE0
0x06AD      0x30E0       MOVLW     224
;PuenteH.c,57 ::          movwf 0x0B
0x06AE      0x008B       MOVWF     INTCON
;PuenteH.c,58 ::          clrf 0x0D //PORTB
0x06AF      0x018D       CLRF      PORTB
;PuenteH.c,59 ::          bsf 0x0D,3 //RB3 = 1 //Enable
```

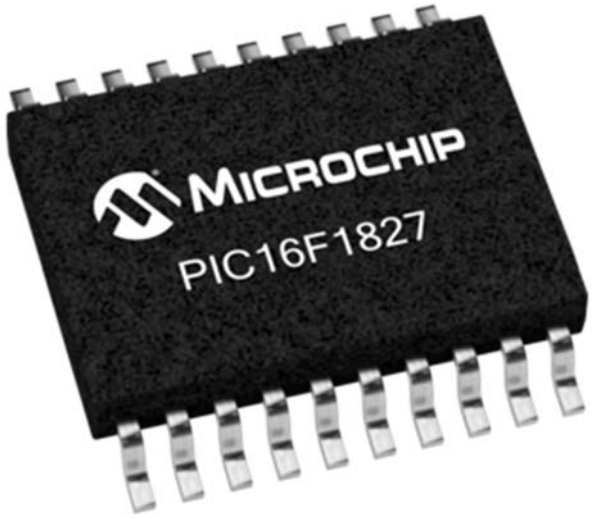
```
0x06B0          0x158D          BSF       PORTB, 3
;PuenteH.c,61 ::                clrf 0x15 //TMR0 = 0
0x06B1          0x0195          CLRF      TMR0
;PuenteH.c,63 ::                }
L_end_Configuracion:
0x06B2          0x0008          RETURN
; end of _Configuracion
_main:
0x06B3          0x26E6          CALL      1766
;PuenteH.c,111 ::               void main() {
;PuenteH.c,112 ::               Configuracion();
0x06B4          0x268F          CALL      _Configuracion
;PuenteH.c,116 ::               tiempo = 61;
0x06B5          0x303D          MOVLW     61
0x06B6          0x0020          MOVLB     0
0x06B7          0x00B2          MOVWF     _tiempo
;PuenteH.c,117 ::               LATB.LATB4 = 1; //Derecha
0x06B8          0x0022          MOVLB     2
0x06B9          0x160D          BSF       LATB, 4
;PuenteH.c,118 ::               LATB.LATB5 = 0;
0x06BA          0x128D          BCF       LATB, 5
;PuenteH.c,119 ::               do{
L_main1:
;PuenteH.c,120 ::               ADC();
0x06BB          0x2621          CALL      _ADC
;PuenteH.c,121 ::               Enviar();
0x06BC          0x2606          CALL      _Enviar
;PuenteH.c,122 ::               if (pot >  0x03FD) {LATB.LATB4 = 0;} //Detenido
0x06BD          0x0020          MOVLB     0
0x06BE          0x0831          MOVF      _pot+1, 0
0x06BF          0x3C03          SUBLW     3
0x06C0          0x1D03          BTFSS     STATUS, 2
0x06C1          0x2EC4          GOTO      L__main14
0x06C2          0x0830          MOVF      _pot, 0
0x06C3          0x3CFD          SUBLW     253
L__main14:
0x06C4          0x1803          BTFSC     STATUS, 0
0x06C5          0x2EC8          GOTO      L_main4
0x06C6          0x0022          MOVLB     2
0x06C7          0x120D          BCF       LATB, 4
L_main4:
;PuenteH.c,123 ::               }while(tiempo);
0x06C8          0x0020          MOVLB     0
0x06C9          0x0832          MOVF      _tiempo, 0
0x06CA          0x1D03          BTFSS     STATUS, 2
0x06CB          0x2EBB          GOTO      L_main1
;PuenteH.c,125 ::               tiempo = 61;
```

```
0x06CC      0x303D      MOVLW    61
0x06CD      0x00B2      MOVWF    _tiempo
;PuenteH.c,126 ::      LATB.LATB4 = 0;
0x06CE      0x0022      MOVLB    2
0x06CF      0x120D      BCF      LATB, 4
;PuenteH.c,127 ::      LATB.LATB5 = 0;
0x06D0      0x128D      BCF      LATB, 5
;PuenteH.c,128 ::      do{
L_main5:
;PuenteH.c,129 ::      ADC();
0x06D1      0x2621      CALL     _ADC
;PuenteH.c,130 ::      Enviar();
0x06D2      0x2606      CALL     _Enviar
;PuenteH.c,132 ::      }while(tiempo);
0x06D3      0x0020      MOVLB    0
0x06D4      0x0832      MOVF     _tiempo, 0
0x06D5      0x1D03      BTFSS    STATUS, 2
0x06D6      0x2ED1      GOTO     L_main5
;PuenteH.c,133 ::      }
L_end_main:
0x06D7      0x2ED7      GOTO     $+0
; end of _main
0x06E6      0x30DB      MOVLW    219
0x06E7      0x0084      MOVWF    FSR0
0x06E8      0x3086      MOVLW    134
0x06E9      0x0085      MOVWF    FSR0H
0x06EA      0x300A      MOVLW    10
0x06EB      0x00F0      MOVWF    R0
0x06EC      0x3001      MOVLW    1
0x06ED      0x00F1      MOVWF    R1
0x06EE      0x3022      MOVLW    34
0x06EF      0x0086      MOVWF    FSR1
0x06F0      0x3000      MOVLW    0
0x06F1      0x0087      MOVWF    FSR1H
0x06F2      0x25FD      CALL     1533
0x06F3      0x0008      RETURN
```

D1.2

Esquemático DAQ

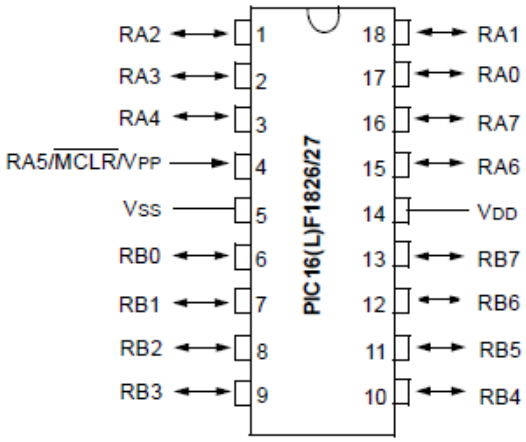
- Características a destacar del microcontrolador PIC16F1827:



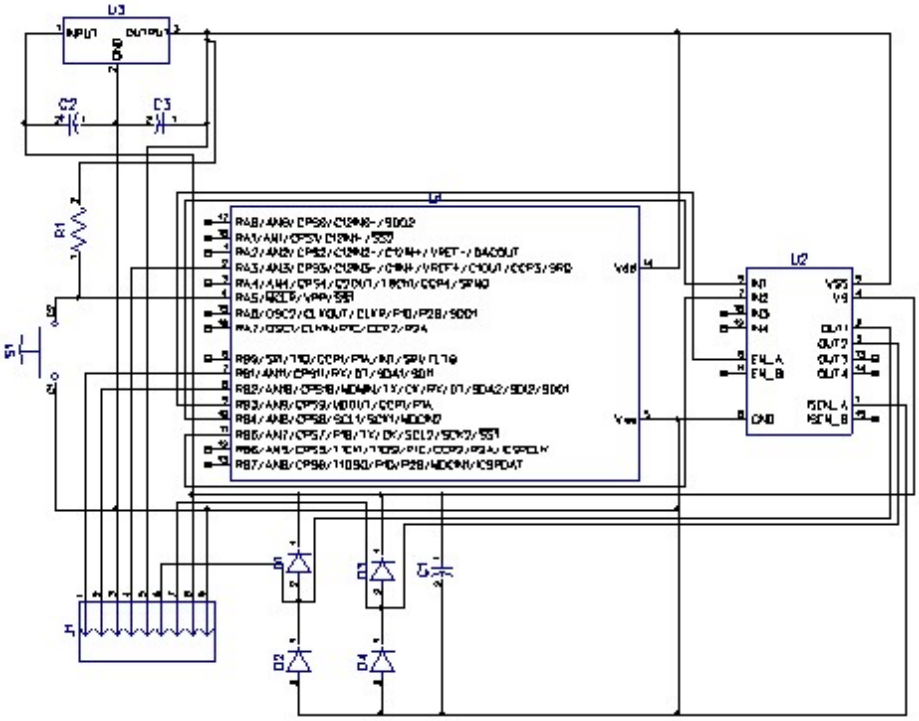
- Convertidor analógico digital de 10 bits
- Oscilador de precisión interno de hasta 32 Mhz
- Memoria Flash: 7kBytes
- Memoria RAM: 384Bytes

Distribución de pines:

PDIP, SOIC



Esquemático circuito DAQ.



D2.1

Firmware control de posición

```
717                     ;psect for function _main
  718  0013                 _main:
  719
  720                   ;incstack = 0
  721                   ; Regs used in _main: [wreg+status,2+status,0+pclath+cstack]
  722  0015                     org      2        ;#
  723
  724                   ;PID.c: 76: TRISA = 0x04;
  725  0015  3004               movlw 4
  726  0016  0021               movlb  1         ; select bank1
  727  0017  008C               movwf12          ;volatile
  728
  729                   ;PID.c: 81: ADCON0=0x09;
  730  0018  3009               movlw 9
  731  0019  009D               movwf29          ;volatile
  732
  733                   ;PID.c: 82: ADCON1 = 0x90;
  734  001A  3090               movlw 144
  735  001B  009E               movwf30          ;volatile
  736
  737                   ;PID.c: 86: DAC1CON0 = 0xA0;
  738  001C  30A0               movlw 160
  739  001D  0022               movlb  2         ; select bank2
  740  001E  0098               movwf24          ;volatile
  741
  742                   ;PID.c: 90: ANSELA = 0x04;
  743  001F  3004               movlw 4
  744  0020  0023               movlb  3         ; select bank3
  745  0021  008C               movwf12          ;volatile
  746
  747                   ;PID.c: 96: PID1CON = 0x85;
  748  0022  3085               movlw 133
  749  0023  002C               movlb  12        ; select bank12
  750  0024  0094               movwf20          ;volatile
  751
  752                   ;PID.c: 104: PPSLOCK = 0x55;
  753  0025  3055               movlw 85
  754  0026  003C               movlb  28        ; select bank28
  755  0027  008F               movwf15          ;volatile
  756
  757                   ;PID.c: 105: PPSLOCK = 0xAA;
  758  0028  30AA               movlw 170
  759  0029  008F               movwf15          ;volatile
```

```
760
761                     ;PID.c: 106: PPSLOCK = 0x00;
762  002A  018F         	clrf	15	;volatile
763
764                     ;PID.c: 107: RC0PPS = 0x08;
765  002B  3008         	movlw	8
766  002C  003D         	movlb	29	; select bank29
767  002D  00A0         	movwf	32	;volatile
768
769                     ;PID.c: 108: RC1PPS = 0x09;
770  002E  3009         	movlw	9
771  002F  00A1         	movwf	33	;volatile
772
773                     ;PID.c: 109: PPSLOCK = 0x01;
774  0030  3001         	movlw	1
775  0031  003C         	movlb	28	; select bank28
776  0032  008F         	movwf	15	;volatile
777
778                     ;PID.c: 111: TRISC = 0x03;
779  0033  3003         	movlw	3
780  0034  0021         	movlb	1	; select bank1
781  0035  008E         	movwf	14	;volatile
782
783                     ;PID.c: 112: PWM3CON = 0x00;
784  0036  002C         	movlb	12	; select bank12
785  0037  0199         	clrf	25	;volatile
786
787                     ;PID.c: 115: PR2=155;
788  0038  309B         	movlw	155
789  0039  0020         	movlb	0	; select bank0
790  003A  009B         	movwf	27	;volatile
791
792                     ;PID.c: 118: PWM3DCH = 0x00;
793  003B  002C         	movlb	12	; select bank12
794  003C  0198         	clrf	24	;volatile
795
796                     ;PID.c: 119: PWM3DCL = 0x00;
797  003D  0197         	clrf	23	;volatile
798
799                     ;PID.c: 121: PIR1bits.TMR2IF = 0;
800  003E  0020         	movlb	0	; select bank0
801  003F  1090         	bcf	16,1	;volatile
802
803                     ;PID.c: 122: T2CLKCON = 0x00;
804  0040  019E         	clrf	30	;volatile
805
806                     ;PID.c: 123: T2RST = 0x0D;
```

```
807  0041  300D          movlw 13
808  0042  009F          movwf 31        ;volatile
809
810                ;PID.c: 124: T2CON = 0xF0;
811  0043  30F0          movlw 240
812  0044  009C          movwf 28        ;volatile
813
814                ;PID.c: 126: PWM3CON = 0x80;
815  0045  3080          movlw 128
816  0046  002C          movlb  12       ; select bank12
817  0047  0099          movwf 25        ;volatile
818
819                ;PID.c: 127: TRISC = 0x00;
820  0048  0021          movlb  1        ; select bank1
821  0049  018E          clrf      14    ;volatile
822
823                ;PID.c: 129: CWG1CON0 = 0x84;
824  004A  3084          movlw 132
825  004B  002D          movlb  13       ; select bank13
826  004C  0096          movwf 22        ;volatile
827
828                ;PID.c: 130: CWG1ISM = 0x09;
829  004D  3009          movlw 9
830  004E  009A          movwf 26        ;volatile
831
832                ;PID.c: 138: PID1K1 = 21870;
833  004F  3055          movlw 85
834  0050  002B          movlb  11       ; select bank11
835  0051  0091          movwf 17        ;volatile
836  0052  306E          movlw 110
837  0053  0090          movwf 16        ;volatile
838
839                ;PID.c: 139: PID1K2 = -36461;
840  0054  3071          movlw 113
841  0055  0093          movwf 19        ;volatile
842  0056  3093          movlw 147
843  0057  0092          movwf 18        ;volatile
844
845                ;PID.c: 140: PID1K3 = 14800;
846  0058  3039          movlw 57
847  0059  0095          movwf 21        ;volatile
848  005A  30D0          movlw 208
849  005B  0094          movwf 20        ;volatile
850
851                ;PID.c: 154: PID1SET = 0x000;
852  005C  018C          clrf      12    ;volatile
853  005D  018D          clrf      13    ;volatile
```

```
854
855                     ;PID.c: 160: PWM3DCH = 0x4E;
856  005E  304E                movlw 78
857  005F  002C                movlb  12       ; select bank12
858  0060  0098                movwf24       ;volatile
859
860                     ;PID.c: 161: PWM3DCL = 0x00;
861  0061  0197                clrf     23       ;volatile
862
863                     ;PID.c: 163: PID1ACCU = 0x00;
864  0062  0193                clrf     19       ;volatile
865
866                     ;PID.c: 164: PID1ACCHH = 0x00;
867  0063  0192                clrf     18       ;volatile
868
869                     ;PID.c: 165: PID1ACCHL = 0x00;
870  0064  0191                clrf     17       ;volatile
871
872                     ;PID.c: 166: PID1ACCLH = 0x00;
873  0065  0190                clrf     16       ;volatile
874
875                     ;PID.c: 167: PID1ACCLL = 0x00;
876  0066  018F                clrf     15       ;volatile
877  0067                  l1103:
878                     ;PID.c: 207: while(1) {
879
880
881                     ;PID.c: 208: _delay((unsigned long)((10)*(8000000/4000.0)));
882  0067  301A                movlw 26
883  0068  0020                movlb  0        ; select bank0
884  0069  00CE                movwf??_main+1
885  006A  30F8                movlw 248
886  006B  00CD                movwf??_main
887  006C                    u1457:
888  006C  0BCD                decfsz  ??_main,f
889  006D  286C                goto    u1457
890  006E  0BCE                decfsz  ??_main+1,f
891  006F  286C                goto    u1457
892  0070  0000                nop
893
894                     ;PID.c: 209: ADCON0bits.GO_nDONE = 1;
895  0071  0021                movlb  1        ; select bank1
896  0072  149D                bsf     29,1    ;volatile
897  0073  189D                btfsc   157,1   ;#
898  0074  2814                goto    ($+-1)  ;#
899
900                     ;PID.c: 212: PID1INH = ADRESH;
```

```
901  0075  0021              movlb 1         ; select bank1
902  0076  081C              movf  28,w      ;volatile
903  0077  002B              movlb 11        ; select bank11
904  0078  008F              movwf15         ;volatile
905
906                    ;PID.c: 213: PID1INL = ADRESL;
907  0079  0021              movlb 1         ; select bank1
908  007A  081B              movf  27,w      ;volatile
909  007B  002B              movlb 11        ; select bank11
910  007C  008E              movwf14         ;volatile
911  007D                 l1113:
912                    ;PID.c: 217: while(PID1CONbits.BUSY)
913
914  007D  002C              movlb 12        ; select bank12
915  007E  1F14              btfss 20,6      ;volatile
916  007F  2890              goto  l1115
917
918                    ;PID.c: 253: error = PID1Z1H;
919  0080  002B              movlb 11        ; select bank11
920  0081  081C              movf  28,w      ;volatile
921  0082  0020              movlb 0         ; select bank0
922  0083  00CD              movwf??_main
923  0084  01CE              clrf  ??_main+1
924  0085  01CF              clrf  ??_main+2
925  0086  01D0              clrf  ??_main+3
926  0087  0850              movf  ??_main+3,w
927  0088  00DC              movwf_error+3
928  0089  084F              movf  ??_main+2,w
929  008A  00DB              movwf_error+2
930  008B  084E              movf  ??_main+1,w
931  008C  00DA              movwf_error+1
932  008D  084D              movf  ??_main,w
933  008E  00D9              movwf_error
934  008F  287D              goto  l1113
935  0090                 l1115:
936
937                    ;PID.c: 254: error = error << 8;
938  0090  0020              movlb 0         ; select bank0
939  0091  0859              movf  _error,w
940  0092  00CD              movwf??_main
941  0093  085A              movf  _error+1,w
942  0094  00CE              movwf??_main+1
943  0095  085B              movf  _error+2,w
944  0096  00CF              movwf??_main+2
945  0097  085C              movf  _error+3,w
946  0098  00D0              movwf??_main+3
947  0099  3008              movlw 8
```

```
948  009A                u1445:
949  009A  35CD            lslf    ??_main,f
950  009B  0DCE            rlf     ??_main+1,f
951  009C  0DCF            rlf     ??_main+2,f
952  009D  0DD0            rlf     ??_main+3,f
953  009E  0B89            decfsz  9,f
954  009F  289A            goto    u1445
955  00A0  0850            movf    ??_main+3,w
956  00A1  00DC            movwf   _error+3
957  00A2  084F            movf    ??_main+2,w
958  00A3  00DB            movwf   _error+2
959  00A4  084E            movf    ??_main+1,w
960  00A5  00DA            movwf   _error+1
961  00A6  084D            movf    ??_main,w
962  00A7  00D9            movwf   _error
963
964                  ;PID.c: 255: error = error + PID1Z1L;
965  00A8  0859            movf    _error,w
966  00A9  00CD            movwf   ??_main
967  00AA  085A            movf    _error+1,w
968  00AB  00CE            movwf   ??_main+1
969  00AC  085B            movf    _error+2,w
970  00AD  00CF            movwf   ??_main+2
971  00AE  085C            movf    _error+3,w
972  00AF  00D0            movwf   ??_main+3
973  00B0  002B            movlb   11      ; select bank11
974  00B1  081B            movf    27,w    ;volatile
975  00B2  0020            movlb   0       ; select bank0
976  00B3  00D1            movwf   ??_main+4
977  00B4  01D2            clrf    ??_main+5
978  00B5  01D3            clrf    ??_main+6
979  00B6  01D4            clrf    ??_main+7
980  00B7  0851            movf    ??_main+4,w
981  00B8  07CD            addwf   ??_main,f
982  00B9  0852            movf    ??_main+5,w
983  00BA  3DCE                    addwfc  ??_main+1,f
984  00BB  0853            movf    ??_main+6,w
985  00BC  3DCF            addwfc  ??_main+2,f
986  00BD  0854            movf    ??_main+7,w
987  00BE  3DD0            addwfc  ??_main+3,f
988  00BF  0850            movf    ??_main+3,w
989  00C0  00DC            movwf   _error+3
990  00C1  084F            movf    ??_main+2,w
991  00C2  00DB            movwf   _error+2
992  00C3  084E            movf    ??_main+1,w
993  00C4  00DA            movwf   _error+1
994  00C5  084D            movf    ??_main,w
```

```
 995  00C6  00D9              movwf _error
 996
 997                    ;PID.c: 256: error = error*4.97;
 998  00C7  085C              movf   _error+3,w
 999  00C8  00FB              movwf___altoft@c+3
1000  00C9  085B              movf   _error+2,w
1001  00CA  00FA              movwf___altoft@c+2
1002  00CB  085A              movf   _error+1,w
1003  00CC  00F9              movwf___altoft@c+1
1004  00CD  0859              movf   _error,w
1005  00CE  00F8              movwf___altoft@c
1006  00CF  3184  2463  3180       fcall    ___altoft
1007  00D2  0878              movf   ?___altoft,w
1008  00D3  0020              movlb  0         ; select bank0
1009  00D4  00A5              movwf___ftmul@f2
1010  00D5  0879              movf   ?___altoft+1,w
1011  00D6  00A6              movwf___ftmul@f2+1
1012  00D7  087A              movf   ?___altoft+2,w
1013  00D8  00A7              movwf___ftmul@f2+2
1014  00D9  300A              movlw 10
1015  00DA  00A2                    movwf___ftmul@f1
1016  00DB  309F              movlw 159
1017  00DC  00A3                    movwf___ftmul@f1+1
1018  00DD  3040              movlw 64
1019  00DE  00A4                    movwf___ftmul@f1+2
1020  00DF  3182  2242  3180       fcall    ___ftmul
1021  00E2  0020              movlb  0         ; select bank0
1022  00E3  0822              movf   ?___ftmul,w
1023  00E4  00BF              movwf___fttol@f1
1024  00E5  0823              movf   ?___ftmul+1,w
1025  00E6  00C0              movwf___fttol@f1+1
1026  00E7  0824              movf   ?___ftmul+2,w
1027  00E8  00C1              movwf___fttol@f1+2
1028  00E9  3182  22F6  3180       fcall    ___fttol
1029  00EC  0020              movlb  0         ; select bank0
1030  00ED  0842              movf   ?___fttol+3,w
1031  00EE  00DC                    movwf _error+3
1032  00EF  0841              movf   ?___fttol+2,w
1033  00F0  00DB              movwf _error+2
1034  00F1  0840              movf   ?___fttol+1,w
1035  00F2  00DA              movwf _error+1
1036  00F3  083F              movf   ?___fttol,w
1037  00F4  00D9              movwf _error
1038
1039                    ;PID.c: 257: error = error/1023;
1040  00F5  3000              movlw 0
1041  00F6  00F3              movwf___aldiv@divisor+3
```

```
1042  00F7  3000            movlw 0
1043  00F8  00F2            movwf ___aldiv@divisor+2
1044  00F9  3003            movlw 3
1045  00FA  00F1            movwf ___aldiv@divisor+1
1046  00FB  30FF            movlw 255
1047  00FC  00F0            movwf ___aldiv@divisor
1048  00FD  085C            movf  _error+3,w
1049  00FE  00F7            movwf ___aldiv@dividend+3
1050  00FF  085B            movf  _error+2,w
1051  0100  00F6            movwf ___aldiv@dividend+2
1052  0101  085A            movf  _error+1,w
1053  0102  00F5            movwf ___aldiv@dividend+1
1054  0103  0859            movf  _error,w
1055  0104  00F4            movwf ___aldiv@dividend
1056  0105  3183  237D  3180       fcall ___aldiv
1057  0108  0873            movf  ?___aldiv+3,w
1058  0109  0020            movlb 0       ; select bank0
1059  010A  00DC                  movwf _error+3
1060  010B  0872            movf  ?___aldiv+2,w
1061  010C  00DB                  movwf _error+2
1062  010D  0871            movf  ?___aldiv+1,w
1063  010E  00DA            movwf _error+1
1064  010F  0870            movf  ?___aldiv,w
1065  0110  00D9            movwf _error
1066
1067                  ;PID.c: 258: PWM3DCH = 15.6*(error)+78;
1068  0111  085C            movf  _error+3,w
1069  0112  00FB            movwf ___altoft@c+3
1070  0113  085B            movf  _error+2,w
1071  0114  00FA            movwf ___altoft@c+2
1072  0115  085A            movf  _error+1,w
1073  0116  00F9            movwf ___altoft@c+1
1074  0117  0859            movf  _error,w
1075  0118  00F8            movwf ___altoft@c
1076  0119  3184  2463  3180       fcall ___altoft
1077  011C  0878            movf  ?___altoft,w
1078  011D  0020            movlb 0       ; select bank0
1079  011E  00A5            movwf ___ftmul@f2
1080  011F  0879            movf  ?___altoft+1,w
1081  0120  00A6            movwf ___ftmul@f2+1
1082  0121  087A            movf  ?___altoft+2,w
1083  0122  00A7            movwf ___ftmul@f2+2
1084  0123  309A            movlw 154
1085  0124  00A2            movwf ___ftmul@f1
1086  0125  3079            movlw 121
1087  0126  00A3            movwf ___ftmul@f1+1
1088  0127  3041            movlw 65
```

```
1089  0128  00A4              movwf   ___ftmul@f1+2
1090  0129  3182  2242  3180        fcall   ___ftmul
1091  012C  0020              movlb 0         ; select bank0
1092  012D  0822              movf   ?___ftmul,w
1093  012E  00B5              movwf   ___ftadd@f2
1094  012F  0823              movf   ?___ftmul+1,w
1095  0130  00B6              movwf   ___ftadd@f2+1
1096  0131  0824              movf   ?___ftmul+2,w
1097  0132  00B7              movwf   ___ftadd@f2+2
1098  0133  3000              movlw 0
1099  0134  00B2              movwf   ___ftadd@f1
1100  0135  309C              movlw 156
1101  0136  00B3              movwf   ___ftadd@f1+1
1102  0137  3042              movlw 66
1103  0138  00B4              movwf   ___ftadd@f1+2
1104  0139  3181  214C  3180        fcall   ___ftadd
1105  013C  0020              movlb 0         ; select bank0
1106  013D  0832              movf   ?___ftadd,w
1107  013E  00BF              movwf   ___fttol@f1
1108  013F  0833              movf   ?___ftadd+1,w
1109  0140  00C0              movwf   ___fttol@f1+1
1110  0141  0834              movf   ?___ftadd+2,w
1111  0142  00C1              movwf   ___fttol@f1+2
1112  0143  3182  22F6  3180        fcall   ___fttol
1113  0146  0020              movlb 0         ; select bank0
1114  0147  083F              movf   ?___fttol,w
1115  0148  002C              movlb 12        ; select bank12
1116  0149  0098              movwf24         ;volatile
1117
1118                    ;PID.c: 259: PWM3DCL = 0x00;
1119  014A  0197              clrf    23      ;volatile
1120  014B  2867              goto    l1103
1121  014C                    __end_of_main:
1122                    ;PID.c: 315: }
1123                    ;PID.c: 316: return;
1124
1125
1126                            psect   text1
1127  02F6                    __ptext1:
1128 ;; *************** function ___fttol *****************
1129 ;; Defined at:
1130 ;;                 line          44          in          file          "C:\Program          Files
(x86)\Microchip\xc8\v1.38\sources\common\fttol.c"
1131 ;; Parameters:    Size  Location     Type
1132 ;;  f1              3   31[BANK0 ] float
1133 ;; Auto vars:     Size  Location     Type
1134 ;;  lval            4   40[BANK0 ] unsigned long
```

```
1135 ;;  exp1            1   44[BANK0 ] unsigned char
1136 ;;  sign1           1   39[BANK0 ] unsigned char
1137 ;; Return value:  Size  Location     Type
1138 ;;                 4   31[BANK0 ] long
1139 ;; Registers used:
1140 ;;                 wreg, status,2, status,0
1141 ;; Tracked objects:
1142 ;;                 On entry : 0/0
1143 ;;                 On exit  : 0/0
1144 ;;                 Unchanged: 0/0
1145 ;; Data sizes:      COMMON    BANK0    BANK1    BANK2    BANK3    BANK4
BANK5   BANK6   BANK7   BANK8   BANK9  BANK10  BANK1
   +1  BANK12
1146 ;;     Params:        0      4      0      0      0      0      0      0      0      0      0      0
   +0       0
1147 ;;     Locals:        0      6      0      0      0      0      0      0      0      0      0      0
   +0       0
1148 ;;     Temps:         0      4      0      0      0      0      0      0      0      0      0      0
   +0       0
1149 ;;     Totals:        0     14      0      0      0      0      0      0      0      0      0      0
   +0       0
1150 ;;Total ram usage:       14 bytes
1151 ;; Hardware stack levels used:    1
1152 ;; This function calls:
1153 ;;                 Nothing
1154 ;; This function is called by:
1155 ;;                 _main
1156 ;; This function uses a non-reentrant model
```

D2.2

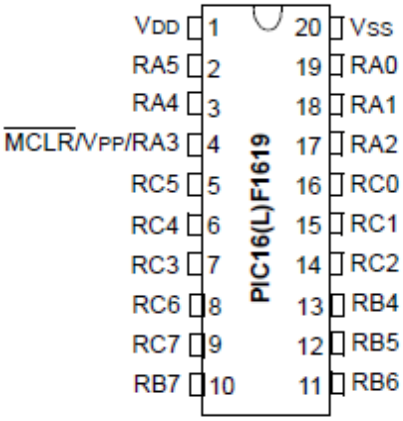
Esquemático del circuito de control de posición

- Características a destacar del PIC16F1619

- Memoria programa: 14KBytes
- Memoria RAM: 1KByte
- Modulo matemático
- Módulo PID
- Oscilador interno de precisión hasta 32MHz
- Módulo CWG
- Módulo PWM con resolución de 10 bits
- Convertidor analógico digital con resolución de 10 bits

Distribución de pines:

20-pin PDIP, SOIC, SSOP



Esquemático circuito control de posición