UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

"GENERADOR DE SEÑALES PERIÓDICAS DE FORMA PREDETERMINADA EMPLEANDO UN DISPOSITIVO CHIPBAS8"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO Y ELECTRÓNICO

P R E S E N T A:

EDUARDO MEZA HERNÁNDEZ

ASESOR:

M.I. ANTONIO SALVÁ CALLEJA

CIUDAD UNIVERSITARIA, FEBRERO DEL 2016.

Índice general

Índice de figuras

Índice de Tablas

1 | Introducción

1.1. Presentación

En los laboratorios de electrónica, un generador de señales es un instrumento indispensable para realizar diversas tareas; aplicar una señal a un circuito para poder observar su respuesta, para reparación y calibración de equipos electrónicos, para investigación y también para fines didácticos.

Estos dispositivos son instrumentos que pueden generar diversos tipos de señales eléctricas variables en el dominio del tiempo, cuyas frecuencias pueden ser ajustables en un rango amplio que depende de las características de los elementos empleados en su construcción, que normalmente varían de 0.2 a los 2 $[MHz]$.

Los generadores de señales se podrían clasificar de acuerdo con las señales que producen, la frecuencia de las mismas y por algunas otras características especiales. Típicamente, producen ondas de forma cuadrada, senoidal y triangular. Sin embargo, existen diferentes tipos de generadores que dependen de la aplicación que se les dará. Por ejemplo, existen algunos para los cuales es posible configurar la forma de onda de una determinada señal periódica a generar, como puede ser el caso de una señal electrocardiográfica que no presenta ninguna de las tres formas básicas mencionadas anteriormente.

Por ende, es conveniente contar con un instrumento que pueda generar señales periódicas con una forma de onda para el periodo básico que puede ser predeterminada. Tal es el caso del dispositivo cuyo diseño y desarrollo se describen en esta tesis: un Generador de Señales Periódicas de Forma Predeterminada (GSPFP).

1.2. Fundamentos teóricos

Una señal se podría definir como una función variable en el tiempo, que transporta datos o algún tipo de información. Para cada instante de tiempo existe un valor único de la función. Existen varios tipos de señales, pero como el objetivo de este trabajo es generar señales periódicas, únicamente se tratarán éstas de forma breve.

Una señal periódica es aquella que completa un patrón dentro de un periodo determinado y repite el mismo patrón en periodos idénticos subsecuentes. El periodo es el tiempo que se tarda en completar el patrón y cada que se completa un patrón se dice que se completa un ciclo. Los ejemplos más comunes de este tipo de señales son aquellas regidas por funciones trigonométricas, como la que se muestra en la figura 1.1. El GSPFP será capaz de generar señales de este tipo pero con la ventaja que se podrá generar casi cualquier forma de onda que el usuario defina.

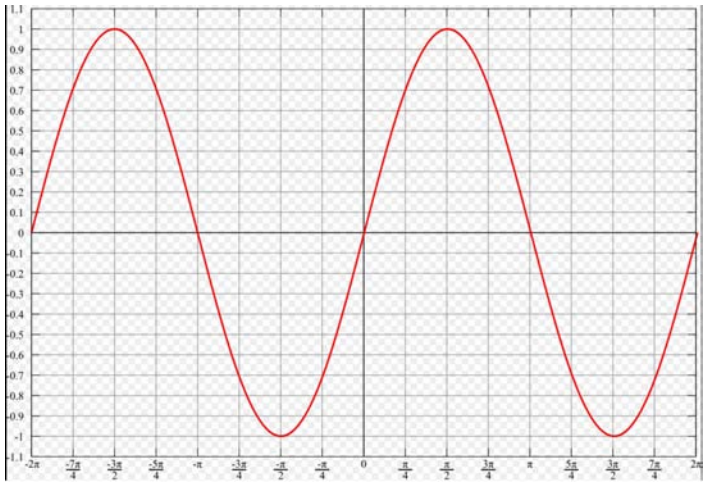

Figura 1.1: Ejemplo de una señal periódica (senoidal).

Es importante mencionar los parámetros generales que describen a las señales que los generadores producen. Las principales características que tiene una señal son la forma de onda, frecuencia y amplitud. La forma de onda es la forma geométrica que tiene la señal, pudiendo ser desde las formas más comunes como la forma senoidal, cuadrada, triangular y diente de sierra, hasta formas de onda muy complejas.

La frecuencia es el número de veces que se produce una onda por unidad de tiempo, o de otra manera, es el recíproco del periodo de la onda. Las unidades de esta magnitud son los hertz.

La amplitud de la señal se refiere a su tamaño, que en términos eléctricos equivale a su nivel de voltaje. Las unidades con las que se mide esta magnitud son los volts. Los generadores tienen diferentes rangos de amplitud según sea la aplicación que se requiera, el GSPFP puede generar señales de hasta 10 $[Vpp]$ (Volts pico a pico).

Otro parámetro importante que todos los generadores tienen es el nivel de offset. Este nivel es el voltaje de corriente directa sobre el cual está montada la señal, por ejemplo si se aplica un offset de 3 [V] significa que la señal generada será desplazada hacia arriba esos 3 [V], o bien, que su componente de directa es 3 [V].

1.3. Planteamiento del problema

Actualmente se han desarrollado modelos matemáticos para generar señales periódicas con forma de onda predeterminada que pueden ser utilizadas en diferentes ramas de investigación. Por ejemplo, existen modelos para generar señales de electrocardiograma, que requieren de un alto costo computacional. Sistemas como el proyectado en esta tesis existen en el mercado, tanto para uso didáctico como industrial. El sistema a desarrollarse en esta tesis produciría un dispositivo generador de señales de periodo predefinido de bajo costo.

Desde luego las frecuencias de las señales a generar no serían tan grandes como las propias de dispositivos comerciales; sin embargo para fines de aplicaciones didácticas y otras tales como biomédicas no se requieren frecuencias muy altas.

1.4. Objetivos

1. Diseñar y construir un sistema para la generación de señales periódicas de forma predeterminada (GSPFP). El sistema estará integrado por una computadora PC ligada a una arquitectura basada en un dispositivo Chipbas8, éste a su vez estará ligado con un convertidor digital-analógico y hardware adicional cuya salida será la señal generada por

el generador de señales objeto de esta tesis.

2. Diseñar una interfaz de usuario (IU) ejecutable en una computadora PC, en la cual se podrá cargar o incluso dibujar la señal periódica deseada, definir el periodo de la señal y modificar en tiempo real algunos parámetros importantes en los dispositivos generadores de señales, como la amplitud de la onda y el nivel de offset.

Para lograr los objetivos antes mencionados se emplearon herramientas de software y hardware para diseño y aprendizaje de sistemas basados en microcontroladores que han sido diseñadas en el Departamento de Control y Robótica de la Facultad de Ingeniería.

1.5. Resultados esperados

El resultado esperado del presente trabajo es la construcción de un prototipo funcional del Generador de Señales Periódicas de Forma Predeterminada. El GSPFP está integrado por los siguientes dos bloques funcionales:

- Interfaz de Usuario (IU)
- Interfaz de Hardware (IHGSPFP)

La IU es una computadora de tipo PC donde corre un software manejador que facilita al usuario final la definición de las características de la señal a generar. Para estos fines, en la PC se ejecuta un programa que valida la interfaz gráfica. El enlace entre la PC y la IHGSPFP es serie asíncrono con una velocidad de 9600 bps y el formato 8N1 (un bit de start, 8 bits de datos y un stop bit). En la figura 1.2 se muestra un esquema del sistema a desarrollar.

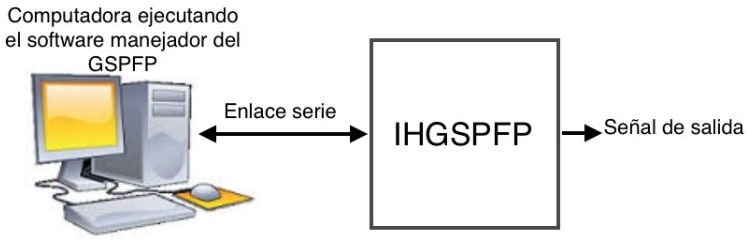

Figura 1.2: Bloques funcionales del GSPFP.

A su vez, la IHGSPFP está constituida por los siguientes bloques:

- Tarjeta MINICON_08GT.
- Etapa de adecuación analógica (EAA).

En la figura 1.3 se muestran los bloques funcionales que constituyen a la IHGSPFP.

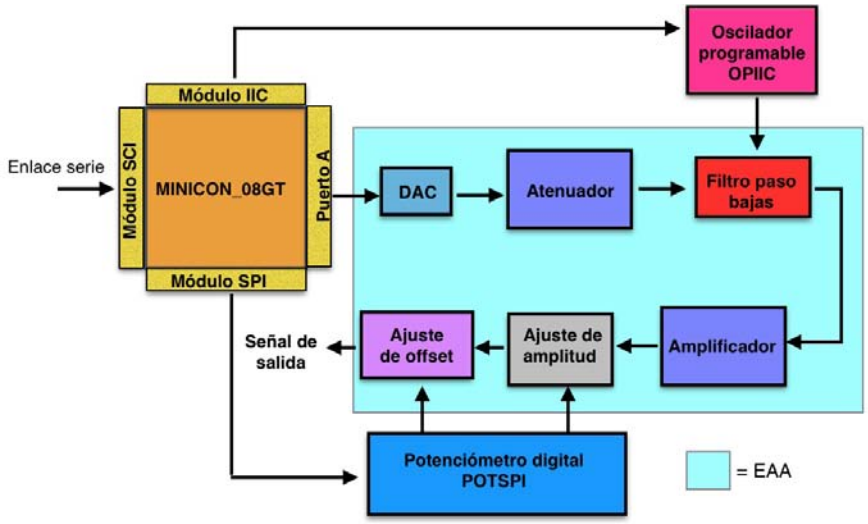

Figura 1.3: Interfaz de Hardware del GSPFP

La tarjeta de desarrollo MINICON_08GT es la encargada de recibir los datos de la computadora por puerto serie, desplegar la señal y controlar algunos bloques externos a través de diversos periféricos de comunicación integrados en el MCU, que son: Módulo SCI, módulo IIC y módulo SPI.

La EEA está compuesta por diferentes bloques, cada uno ellos realiza una función específica para la adecuación de la señal conforme lo requiera el usuario. Éstos se detallarán a continuación:

DAC: es el convertidor digital-analógico que se encargará de convertir los datos binarios que salen por el puerto A de la tarjeta a una señal analógica. A la salida del DAC la amplitud de la señal será de 20 [Vpp].

Atenuador: es un circuito cuya función es atenuar la señal de la salida del DAC a un nivel de 4 [Vpp], de modo que el nivel de la señal se encuentre dentro del rango aceptable por el filtro del siguiente bloque.

Filtro paso bajas: esta etapa se requiere para para suavizar las discontinuidades finitas que presenta la salida del DAC. Para la realización de esta etapa se utilizó un filtro de capacitancia conmutada, validado por el circuito integrado LMF100 de Texas Instruments. Para el funcionamiento del filtro se requiere de una señal de reloj cuya frecuencia es un múltiplo de la frecuencia de corte del filtro. Esta señal de reloj es generada por un oscilador programable con interfaz IIC (OPIIC). La configuración requerida para el OPIIC en un momento dado es hecha desde el MCU por medio de la interfaz IIC de éste.

Amplificador: este bloque es un circuito amplificador con ganancia 2.5, de este modo el valor pico a pico máximo de la salida del GSPFP será 10 [Vpp], que es la amplitud máxima que el GSPFP podrá generar.

Ajuste de amplitud: aquí se empleó un circuito cuya función es ajustar la amplitud de la señal de acuerdo con el nivel que el usuario establezca en la IU. El valor de amplitud que puede tomar está comprendido en un rango de 0 a 10 [Vpp].

Ajuste de offset: corresponde a un circuito para configurar el nivel de offset que el usuario defina previamente en la IU. El valor del nivel de offset está comprendido en un rango de -5 a 5 [V].

Para los circuitos en los que se ajusta la amplitud y el offset se empleó un potenciómetro controlado digitalmente por el MCU a través del bus SPI.

A la salida del GSPFP se deberá tener la señal con la forma de onda, periodo, amplitud y nivel de offset que el usuario haya definido previamente en la IU.

1.6. Organización del trabajo

En el capítulo 2, se presentan las características principales del ambiente de desarrollo del presente trabajo de tesis, compuesto por diferentes elementos de software y hardware.

El capítulo 3 contiene el desarrollo del software que se ejecuta en el microcontrolador, que contiene diagramas de flujo de las rutinas necesarias para lograr desplegar una señal a través de uno de sus puertos digitales. Además se presenta una breve explicación de los buses de comunicación seriales utilizados para la comunicación del microcontrolador con dispositivos externos.

En el capítulo 4 se explican las funcionalidades que brinda la interfaz de usuario desarrollada, en la que se definen las características principales de una señal deseada.

El capítulo 5 habla sobre el diseño de los circuitos digitales y analógicos necesarios para desplegar y acondicionar la señal que se define en la interfaz de usuario. Además, se explica el funcionamiento de algunos circuitos integrados utilizados.

En el capítulo 6, se muestran las pruebas del funcionamiento del prototipo creado, para observar la señal resultante tras cada etapa de hardware.

En el capítulo 7 se tratan algunos ejemplos de aplicación directa del GSPFP.

Por último, en el capítulo 8, se presentan las conclusiones generales de la tesis.

Al final también se anexan los códigos de los programas ejecutables en el MCU de la tarjeta MINICON_08GT y los propios de la interfaz de usuario que corre en la PC.

2 Descripción y funcionalidad de un dispositivo CHIPBAS8 y tarjeta MINICON_08GT para desarrollo asociada con éste

Los microcontroladores están presentes en cualquier lugar; en el trabajo, en casa, en la industria, en el espacio y en general, en nuestra vida. Pueden estar controlando desde la temperatura de un horno en casa, nuestro teléfono celular y hasta algún proceso industrial. Cada vez dependemos más de ellos, sin ellos no serían posibles las comunicaciones, no habría producción a grandes escalas en las fábricas, no tendríamos equipos electrónicos utilizados en hospitales, ni tampoco sistemas que nos facilitan la vida a diario como las lavadoras y los hornos, o simplemente no tendríamos un sistema que se ha vuelto esencial para el desarrollo de la humanidad: la computadora.

Debido a la gran importancia y a la amplia gama de aplicaciones que se le puede dar a los microcontroladores existen varios fabricantes de estos dispositivos. El que se tratará específicamente en esta sección es un microcontrolador de la familia HCS08 de Freescale Semiconductor.

2.1. Acercamiento a los microcontroladores

Un microcontrolador es un circuito integrado de alta escala de integración que dentro de un solo chip contiene una computadora digital y periféricos necesarios para una aplicación real. Este dispositivo es capaz de ejecutar un programa grabado previamente en su memoria.

A diferencia de un microprocesador, un microcontrolador requiere de un número mínimo de chips externos de apoyo para su funcionamiento. Por ejemplo, un microcontrolador tendrá una memoria RAM/ROM/FLASH, un

convertidor analógico-digital, temporizadores y buses de interfaces de comunicación.

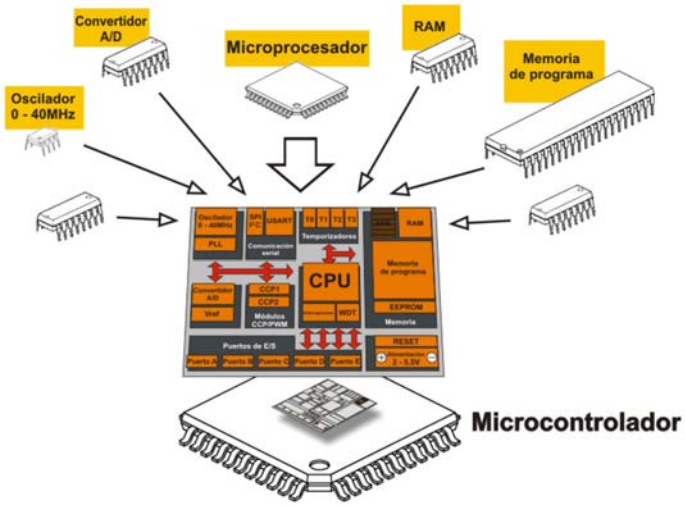

Figura 2.1: Representación de un microcontrolador.

Los microcontroladores son usualmente llamados microcomputadoras, pues contienen las tres unidades funcionales de una computadora: una unidad central de procesamiento, memoria y unidades de entrada/salida.

El CPU (Central Processing Unit) es el elemento central de cualquier microcontrolador, podría definirse como el cerebro donde la mayoría de los cálculos tienen lugar. Dentro del CPU se encuentra la ALU (Arithmetic Logic Unit), que realiza las operaciones aritméticas y lógicas y también se encuentra la Unidad de Control, la cual extrae las instrucciones de la memoria y las decodifica para ejecutarlas.

2.2. El microcontrolador MC9S08GT60A

El MC9S08GT60 es un microcontrolador de bajo costo y de alto rendimiento, que pertenece a la familia HCS08. El tamaño de los registros internos de este MCU es de ocho bits, lo que significa que este número de bits pueden ser procesados simultáneamente. De acuerdo con la hoja de datos [1], tiene las siguientes características:

- 60 KBytes de memoria Flash programable.

- 4 KBytes de memoria RAM.
- 33 pines de propósito general para entradas/salidas digitales.
- Múltiples opciones de fuente de reloj: generador interno, cristal o reloj externo.
- Resistencias de pull up programables en los puertos cuando funcionan como entradas.
- Soporta hasta 32 fuentes de interrupción.

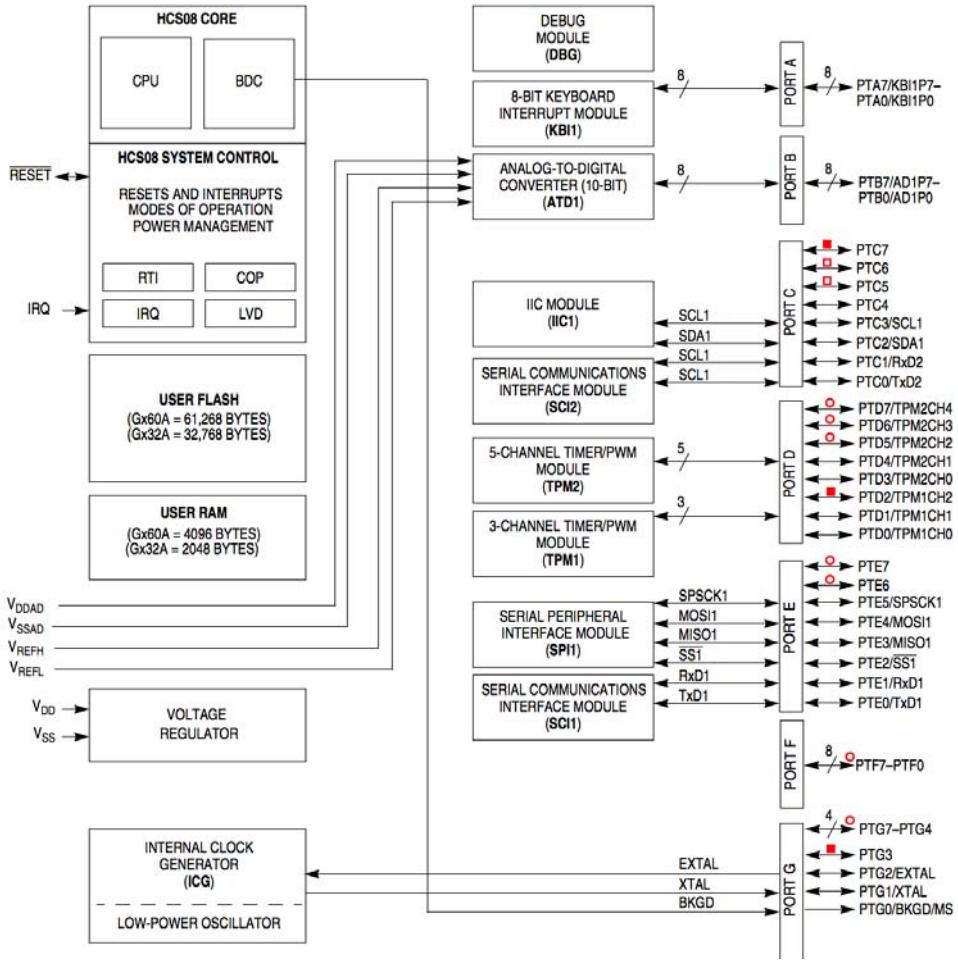

Figura 2.2: Diagrama de bloques del MCU MC9S08GT60A

En la figura 2.2 se aprecian los bloques funcionales del MCU, así como los módulos y periféricos con sus respectivas conexiones a los puertos. La explicación de la ubicación de los diferentes módulos se abordará en la sección de la tarjeta MINICON_08GT, pues ésta cuenta con los pines correspondientes a cada puerto. La siguiente tabla resume los módulos presentes:

Módulo	**Cantidad presente**
Convertidor analógico - digital (CAD)	1
Generador interno de reloj	1
Interfaz de periféricos serie (SPI)	1
Interfaz de comunicación serial (SCI)	2
Circuitos inter - integrados (IIC)	1
Interrupción por teclado (KBI)	1
Modulación de ancho de pulsos (PWM)	2

Tabla 2.1: Módulos presentes en el MCU MC9S08GT60A

2.2.1. Unidad de procesamiento central S08CPUV2

Es el elemento central del MC9S08GT60A, en esta sección se mencionarán las principales características del CPU S08CPUV2:

- Registro de pila (Stack pointer) de 16 bits.
- Registro índice H:X (Index register) de 16 bits.
- Acumulador A de 8 bits.
- Siete modos de direccionamiento: Inherente,relativo, inmediato, directo, extendido, relativo a H:X y relativo al Stack Pointer.
- Registro que contiene banderas de condiciones: acarreos, resultado cero, restado negativo y desbordamiento.

La plataforma de software y hardware que se empleó para el desarrollo de este trabajo de tesis fue el sistema AIDA08 para desarrollo y aprendizaje con microcontroladores de la familia HC(S)08 de Freescale. Este sistema de desarrollo se diseñó y construyó en el Departamento de Control y Robótica de la Facultad de Ingeniería. A continuación se describe lo general el sistema AIDA08.

2.3. El sistema AIDA08

El sistema AIDA08 (Ambiente Integrado para Desarrollo y Aprendizaje con microcontroladores de la familia HCS08 de Freescale) es una herramienta mediante la cual se pueden desarrollar aplicaciones para los microcontroladores de Freescale, así como prueba inmediata de software en la tarjeta MINICON_08GT escrito en ensamblador y/o BASIC. Este sistema está integrado por dos bloques funcionales de software y hardware:

A) Tarjeta de desarrollo MINICON_08GT ligada a una computadora con un software manejador, mediante una conexión serie.

B) Software manejador llamado PUMMA_EST, el cual cuenta con un ensamblador cruzado llamado ENS08 y un compilador cruzado de Basic denominado MINIBAS8A.

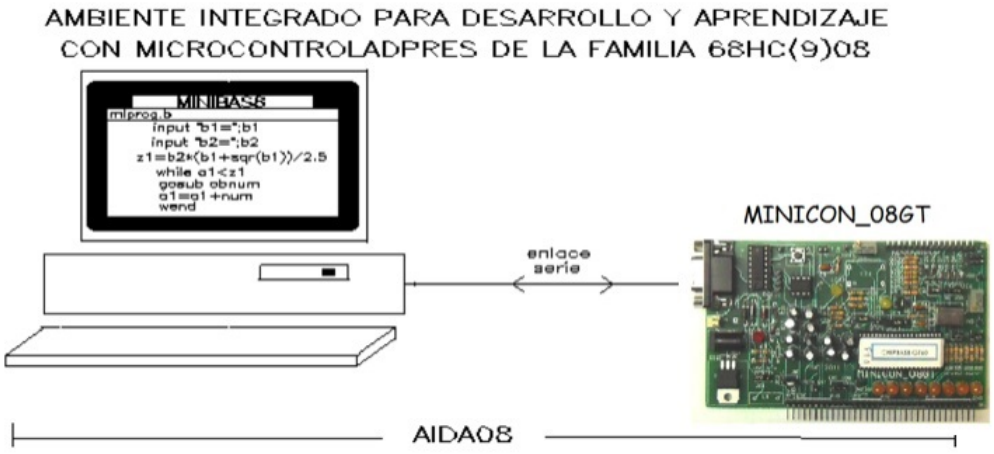

Figura 2.3: Componente funcionales del sistema AIDA08

2.3.1. Dispositivos CHIPBAS8

Un microcontrolador de la familia HC08 de Freescale que contiene en su memoria el firmware base NBCP8_BIBAS8 es considerado un dispositivo CHIPBAS8. Dicho firmware está conformado por las bibliotecas del compilador MINIBAS8A (BIBAS8) y de un enrutador y receptor de comandos (NBCP8), este último habilita que el MCU pueda ser manejado desde una

PC donde se ejecuta un software manejador denominado PUMMA_EST.

La designación de los dispositivos CHIPBAS8 llevan un posfijo que denota el tipo de MCU de la familia HC(S)08 implicada. Para este trabajo de tesis se empleó un dispositivo CHIPBAS8GT que está basado en el MCU MCS08GT60 de Freescale.

2.3.2. Software manejador PUMMA_EST

Como se mencionó anteriormente, este es el software encargado de establecer la comunicación entre la computadora y la tarjeta de desarrollo, para el manejo de esta última. El software PUMMA_EST es la última versión de este concepto de software, que abarca también facilidades para desarrollo con microcontroladores MSP430 de Texas Instruments. Entre las principales características de este software se encuentran las siguientes:

- Capacidad para escribir datos a memoria o a un puerto de la tarjeta.
- Capacidad para examinar memoria en la tarjeta.
- Capacidad para programar memoria FLASH contenida en el MCU.
- Contiene un editor básico para que el usuario pueda escribir y almacenar los programas.
- Contiene un ensamblador cruzado, denominado ENS08.
- Contiene un compilador cruzado de lenguaje BASIC, denominado MINIBAS8.
- Capacidad para obtener, a partir del código fuente en ensamblador, el archivo S19 que contiene código máquina ejecutable en el MCU.
- Capacidad para obtener, a partir del código fuente en BASIC, el archivo S19 que contiene código máquina ejecutable en el MCU.
- Capacidad para ejecutar de inmediato el código generado a partir de un determinado programa fuente escrito en el lenguaje ensamblador o BASIC.
- Contiene un emulador de terminal básico mediante el cual se realiza la consola de interfaz con el usuario para fines de la ejecución de programas en BASIC.

2.4. La tarjeta MINICON_08GT

La tarjeta MINICON_08GT es el bloque central del GSPFP que administra y controla los elementos funcionales de entrada y salida presentes en el GSPFP. Como núcleo central de funcionamiento tiene al microcontrolador MC9S08GT60A, con firmware de base que lo habilita como dispositivo CHIPBAS8GT. A continuación se mencionan las principales características de la tarjeta:

- Basada en el microcontrolador MC9S08GT60A fabricado por FREESCALE.
- Contiene postes con diversos accesos a puntos de interés, como líneas de entrada o salida de puertos y diversos puntos de prueba.
- Incluye LEDs testigo para el puerto A, que sirven como auxiliar didáctico.
- Contiene interfaz para desplegados alfanuméricos populares en la industria.
- Contiene pines de cinco puertos (A, B, C, D, E) que pueden ser configurados como pines de propósito general (entradas y salidas digitales) o se pueden habilitar módulos integrados para realizar una función específica.
- Capacidad para la ejecución autónoma de programas cargados en la memoria FLASH con facilidades propias del manejador PUMMA_08.

La imagen de la figura 2.4 presenta a la MINICON_08GT dividida en cuatro cuadrantes que facilitará encontrar la posición de los postes y conectores presentes.

La tabla 2.2 muestra la ubicación y el uso de los nueve postes de referencia presentes en la tarjeta MINICON_08GT.

La tabla 2.3 muestra la posición y el uso de los siete jumpers excluyentes presentes en la MINICON_08GT.

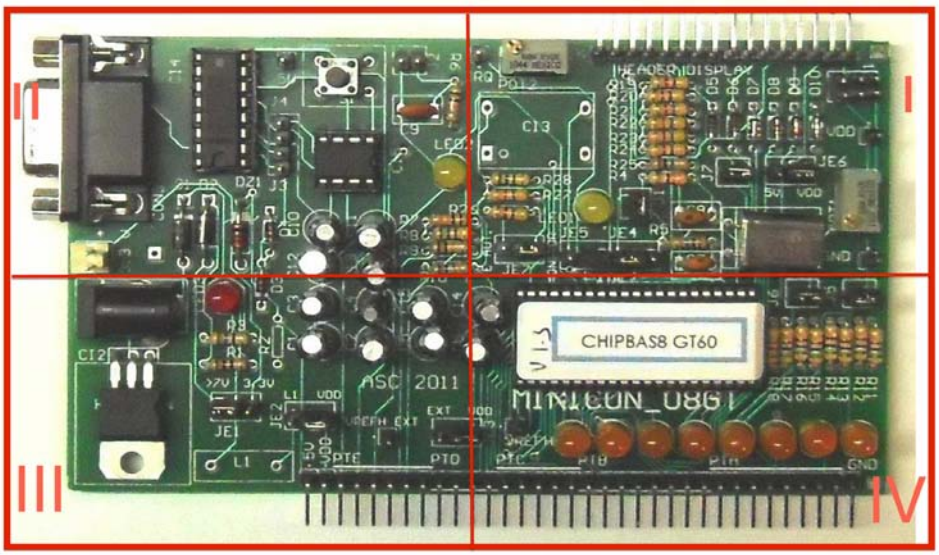

Figura 2.4: División en cuadrantes de la MINICON_08GT

Nombre	**Ubicación**	**Posición a la izquierda**	**Posición a la derecha**
JE1	III	Habilita alimentación por el conector de barril, usando el regulador de 3.3V	Habilita alimentación por el header de 3.3V
JE2	III	Habilita el uso del inductor L1	Conecta la fuente del DAC a Vdd
JE3	III	Habilita el uso de un voltaje de referencia externo	Usa Vdd como VREF
JE4	I	Habilita el uso del reloj oscilador Xtal	Habilita el uso del módulo oscilador CI3
JE5	I	Habilita el uso del módulo oscilador CI3	Habilita el uso del reloj oscilador Xtal
JE6	I	Direcciona 5V como alimentación al Display	Direcciona 3.3V como alimentación al Display
JE7	I	Habilita el modo de ejecución autónoma	Habilita el modo de ejecución monitor

Tabla 2.3: Jumpers excluyentes de la MINICON_08GT

Poste	Ubicación	Funcionalidad
5V	II	Testigo del bus de 5V
5V H	III	Testigo del bus de 5V
GND	I	Testigo del bus de tierra
GND H	IV	Testigo del bus de tierra
IRQ	I	Poste de acceso a interrupción IRQ
VDD	I	Testigo del bus de 3.3V
VDD H	III	Testigo del bus de 3.3V
VREFH	IV	Testigo del nivel de referencia externo
VREFH EXT	III	Poste para ingresar un valor de referencia externo

Tabla 2.2: Postes de referencia de la MINICON_08GT

2.4.1. Pines de entradas y salidas paralelas

En esta sección se explicará a grandes rasgos la utilidad en el proyecto de los pines de entrada y/o salida que tiene la tarjeta. Todos los pines de los puertos pueden funcionar como pines de propósito general y en cada uno de ellos se puede configurar por software la resistencia de pull up. Muchos de estos pines son compartidos por periféricos internos del MCU como temporizadores, interrupciones externas o módulos de comunicación. Para mencionar la función y localización de los módulos en los pines de la tarjeta es necesario estudiarlos por el puerto en el que se encuentran.

Puerto A

El puerto A es un puerto compartido de 8 bits que puede ser configurado para recibir interrupciones por teclado a través del módulo KBI (Keyboard Interrupt), o bien, como un puerto de propósito general en entradas y salidas. En el desarrollo del trabajo el puerto A es utilizado como puerto de propósito general usando los pines como salidas digitales. La señal que entrega el GSPFP es generada a través de las salidas digitales del puerto A, convertida a una señal analógica por medio de un DAC (Convertidor Digital-Analógico), explicado posteriormente en el trabajo.

Puerto B

El puerto B funciona también como un puerto compartido de 8 bits, pudiendo ser configurado como un convertidor Analógico-Digital o como puerto

de propósito general de entradas y salidas digitales. Este puerto no es utilizado en el desarrollo del GSPFP.

Puerto C

Este puerto es compartido por los módulos SCI2 (Serial Communication Interface 2), IIC (Inter Integrated Circuit) y por un módulo de entradas y salidas digitales. Presenta gran importancia para llevar a cabo el presente trabajo, pues el puerto C es habilitado para trabajar con el módulo IIC, que servirá para establecer comunicación con un módulo oscilador programable.

Puerto D

Los pines del puerto D son compartidos por dos módulos independientes PWM (Pulse Width Modulation) y nuevamente puede funcionar como puerto de entradas y salidas digitales. Este puerto no tiene efecto en el GSPFP.

Puerto E

Es un puerto compartido por tres módulos, uno de entradas y salidas digitales, el módulo SCI1 (Serial Communication Interface 1) y el módulo SPI (Serial Peripheral Interface). Este último es habilitado para establecer comunicación con un potenciómetro digital programable. La funcionalidad y descripción de éste se tocará en el capítulo 5.

3 Desarrollo del software ejecutable en el dispositivo CHIPBAS8

En este capítulo se explicará a detalle el funcionamiento del software y las rutinas que se desarrollaron con el sistema AIDA08, detallado en el anterior capítulo. Gracias a las cualidades del software PUMMA_EST, que entre sus funcionalidades contiene al ensamblador cruzado ENS08 y al compilador cruzado de BASIC denominado MINIBAS8A, el software ejecutable en el MCU de la tarjeta MINICON_08GT desarrollado para fines de esta tesis, está escrito en lenguaje BASIC con incrustaciones en lenguaje ensamblador, lo cual nos brinda la posibilidad de programar con las facilidades de ambos lenguajes.

Para un mejor entendimiento del programa, se recomienda consultar la hoja de datos del MCU MC9S08GT60 [1]. Dicho manual contiene el set de instrucciones en lenguaje ensamblador, la dirección de memoria de los registros internos del MCU y otros datos importantes que ayudarán a la comprensión del código. Con el objeto de comprender la sintaxis de los lenguajes ensamblador y BASIC empleados pueden verse respectivamente los capítulos 5 y 7 del manual de AIDA08 [2].

Además se estudiará con detalle el funcionamiento de los módulos o periféricos de comunicación integrados en el MCU, que son utilizados para comunicarse con dispositivos utilizados en el proyecto. Estos módulos son: módulo de la interfaz de comunicación serie (SCI), módulo de interfaz de periféricos serie (SPI) y el módulo de comunicación de circuitos inter - integrados (IIC).

3.1. Generación de una señal a partir del programa del MCU

Para crear una idea de cómo es que el GSPFP genera una señal, en esta sección se explicará de modo muy general la forma en que se obtiene una

señal a partir del programa ejecutable en el MCU. Posteriormente en este mismo capítulo mismo se explicará a detalle el programa del MCU y en el siguiente capítulo el software ejecutable en la PC, pues en su conjunto es como se logra obtener la señal.

El software ejecutable en la PC es el encargado de obtener todos los datos correspondientes a la señal periódica que se desea desplegar en el GSPFP, así como los datos de la amplitud y el nivel de offset de la señal. Una vez teniendo esos datos, byte por byte son transferidos al MCU y este último los procesa para poder desplegar la señal digital a través del puerto A, que a su vez será la empleada como entrada al convertidor digital-analógico (DAC).

3.2. Estructura general del programa ejecutable en el MCU para fines del GSPFP

Ahora se explica el algoritmo general del funcionamiento del programa que se encuentra en la memoria del MCU para lograr generar una señal del GSPFP. La figura 3.1 representa el diagrama de flujo que demuestra el funcionamiento general del programa, que se puede resumir en lo siguiente:

1. Configuración de las direcciones de memoria de los registros a usar.
2. A través del puerto serie se capturan los datos de la señal: número de muestras por periodo, periodo entre muestras y valor del byte que representa cada muestra.
3. A través de puerto serie se capturan los bytes que serán enviados al oscilador programable y se le envían a éste por el módulo IIC.
4. Se recibe un byte más, el cual indicará el valor del preescalador a utilizar en la interrupción por tiempo programable. Esta última se activa. La rutina de interrupción es la encargada de colocar los valores de cada muestra en el puerto A.
5. Entra a un loop (lazo) en el cual se queda a la espera de un byte que indica que acción ejecutar. Si ese byte es igual a cero, se capturan los valores para configurar a los potenciómetros digitales que ajustan la amplitud y nivel de offset de la señal. Estos valores son enviados a los potenciómetros digitales a través del módulo SPI. Si el byte recibido es igual a uno, se ejecuta un comando para quitar la señal del GSPFP, limpiar la interrupción y poner al MCU en el modo recepción de comandos.

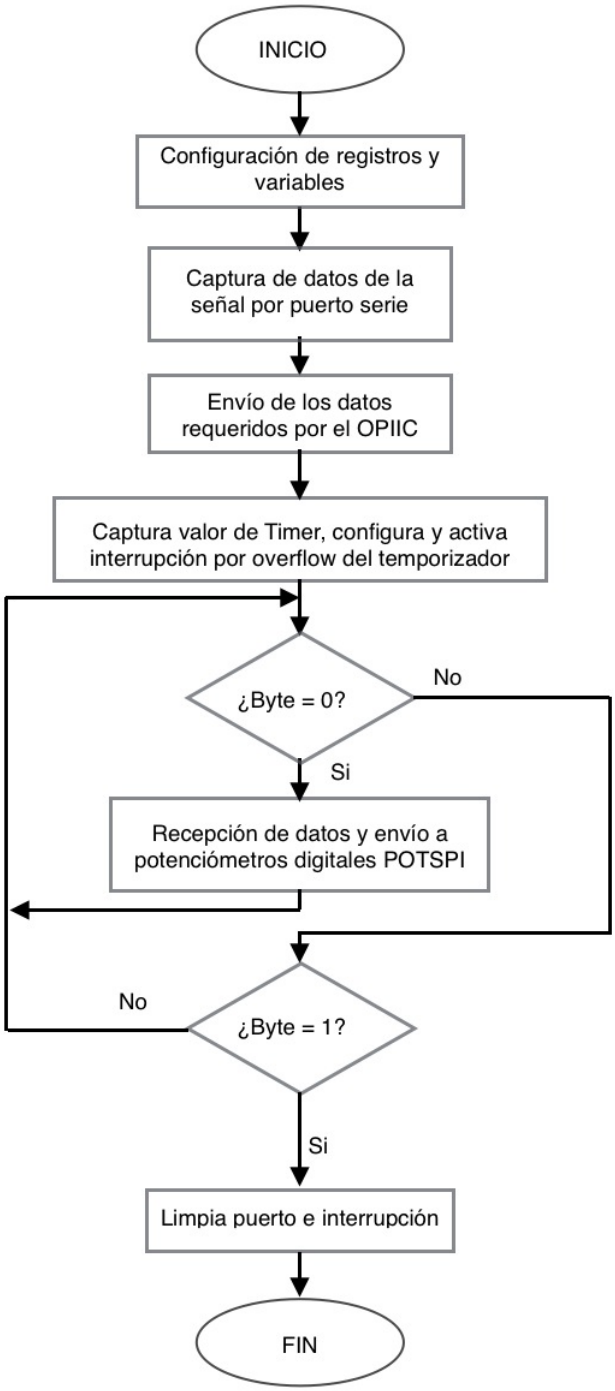

Figura 3.1: Diagrama de flujo con la estructura general del programa

Cada bloque de la estructura general del programa se detallará a continuación.

3.3. Configuración de registros y variables

En este bloque del programa se definen las variables y las direcciones de memoria de los registros a utilizar, cada registro forma parte de un conjunto de registros que controlan el funcionamiento de algún módulo interno del MCU. Las direcciones de memoria de todos los registros del MCU se pueden consultar en el capítulo 4 de su hoja de datos [1].

En el programa se usan múltiples variables, sin embargo son declaradas de forma implícita. La variable *byacad* es declarada de forma explícita como un arreglo de enteros, el cual contendrá la tabla de los valores de las muestras a colocar en el DAC.

```
dim byacad(1024) as integer
```

Se definen las direcciones de memoria de los registros que serán manipulados con código de lenguaje BASIC:

```
defvarbptr ptadd &h03
defvarbptr spic1 &h28
defvarbptr spic2 &h29
defvarbptr spibr &h2a
defvarbptr iic1a &h58
defvarbptr iic1f &h59
defvarbptr iic1c &h5a
```

También es necesario configurar las direcciones en memoria de los registros que son manipulados en secciones de lenguaje ensamblador:

```
iniens
tpm1sc@ equ $30
ptad@ equ $00
tpm1modh@ equ $33
spis@ equ $2b
spid@ equ $2d
iic1c@ equ $5a
iic1s@ equ $5b
```

iic1d@ equ $5c
iic1f@ equ $59
finens

3.4. Captura de los datos de la señal por puerto serie

Los parámetros esenciales para construir la señal periódica a desplegar son los siguientes: número de muestras por periodo, periodo entre muestras y el arreglo de bytes que se reciclará en el puerto A, de modo que la salida del DAC, cuya entrada es la propia del puerto A, sea la señal deseada, homologada a que sus valores estén en un intervalo comprendido entre -10 y 10 [V].

Los valores de los parámetros mencionados son determinados en el software que corre en la computadora, enviados por ésta y capturados por la tarjeta MINICON_08GT a través del puerto serie. La mayoría de las PC cuentan con una conexión DB9 macho para la comunicación serie, sin embargo si no se cuenta con tal puerto como es el caso de las Laptops, se puede usar un puerto USB como interfaz de puerto serie. Las pruebas para probar la comunicación serie de este proyecto se hicieron con un cable convertidor de USB a DB9, conectados a una laptop y a la tarjeta de desarrollo respectivamente.

Como se utiliza la transferencia de datos por el puerto serie, es necesario detallar el módulo de comunicación serie presente en el MCU para entender el funcionamiento del protocolo de comunicación.

3.4.1. SCI Interfaz de comunicación serial

Es una interfaz de comunicación asíncrona implementada en el MCU MC9S08GT60 que permite el intercambio de datos con diversos periféricos externos. El MCU en uso contiene dos puertos serie independientes denominados SCI1 y SCI2. En el capítulo 11 de la hoja de datos del MCU [1] se detallan las características del módulo presente, a continuación se listan las principales:

- Capacidad de enviar y recibir datos al mismo tiempo (full duplex).
- Velocidad programable para envío y recepción de datos.
- Operación con interrupciones por diferentes eventos del SCI.

- Longitud de datos programable (8 o 9 bits).

Transmisión de datos

La comunicación es iniciada por un bit de inicio (estado bajo), seguida por los bits de datos (comúnmente 8, pero pueden ser 9), un bit de paridad (opcional) y un bit de paro (estado alto). Este proceso se puede observar en la figura 3.2.

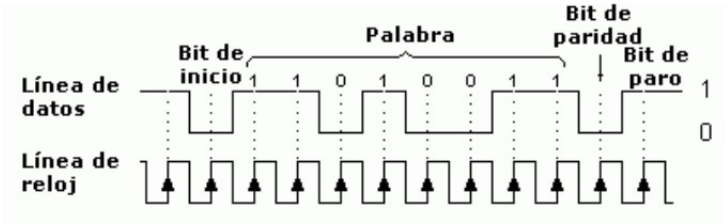

Figura 3.2: Proceso de transmisión por el bus SCI

El principio de operación del módulo SCI es el siguiente: existen dos registros de corrimiento, uno para el transmisor que convierte los datos paralelos a datos de tipo serial (se envía un bit detrás de otro) y otro para el receptor que convierte los datos de tipo serial a datos paralelos. Ambos registros necesitan de una señal de reloj, que es única para los dos registros. Cabe destacar que el receptor y el transmisor trabajan de manera independiente, lo que presenta una ventaja sobre otras interfaces de comunicación serial.

Subrutina de la biblioteca lee#car

Con esta subrutina implementada en la biblioteca de MINIBAS8A, se ahorra tiempo y espacio en la escritura del código, pues ya no se necesita generar una subrutina especial para recibir un byte, en la cual estarían involucrada la configuración de los registros propios del módulo SCI.

Al invocarse esta subrutina entra a un lazo de espera de 1 byte a través del puerto serie. Al retornar dicho byte estará contenido en el acumulador A del MCU.

3.4.2. Código del programa para recibir los datos de una señal por SCI

Para entender este bloque de código se deben definir las variables usadas:

nmp% Variable de tipo *integer* que hace referencia al número de muestras por periodo.

xr& Variable de tipo *long* que almacena el periodo entre muestras.

byacad() Arreglo tipo *integer* que guarda los valores de todas las muestras a colocar en el puerto.

capbte~ Variable tipo *byte* que almacena el valor de un byte recibido por puerto serie.

El diagrama de flujo de la figura 3.3 muestra el proceso que lleva a cabo el bloque de código del programa que recibe los datos de la señal a través del puerto serie.

Como se observa en el diagrama, se recibe byte por byte cada dato de la señal y tras completar el llenado del arreglo que contiene los valores de las muestras, se tienen todos los datos esenciales que forman a la señal.

Primero se declara la variable que almacenará el número de muestras por periodo y se llama a la subrutina que retorna el valor del byte en la variable *capbte ~*. Mediante una incrustación en ensamblador se guarda el valor del byte alto y posteriormente del byte bajo del valor de *nmp%*. El diagrama de flujo llevado a código es el siguiente:

```
nmp%=0
gosub cap_byte
iniens
lda capbte~
sta nmp%
finens

gosub cap_byte
iniens
lda capbte~
sta nmp%+1
finens
```

Mismo procedimiento que el anterior para obtener el valor de los bytes que forman a la variable *xr&*:

```
xr&=0
```

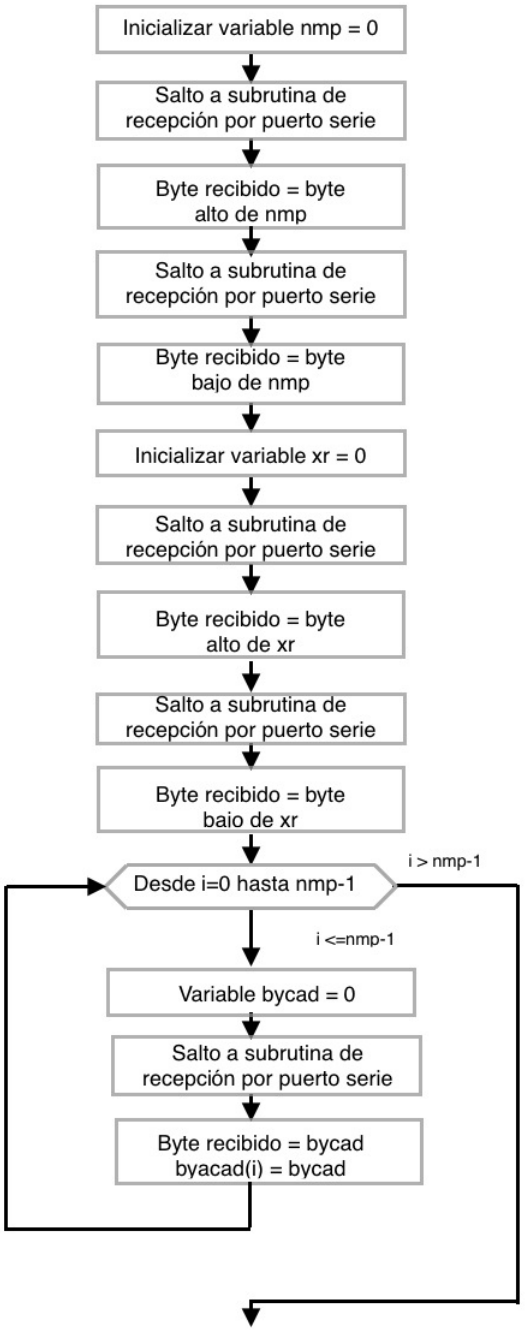

Figura 3.3: Diagrama de flujo para la recepción por SCI de los datos que conforman a la señal

```
gosub cap_byte
iniens
lda capbte~
sta xr&+2
finens

gosub cap_byte
iniens
lda capbte~
sta xr&+3
finens
```

La última sección de código de este bloque es la captura del arreglo de bytes que contiene los valores de cada muestra, logrado fácilmente con tramos de código en ambos lenguajes que el software PUMMA_EST permite manejar:

```
for i%=0 to nmp%-1
bycad%=0
gosub cap_byte
iniens
lda capbte~
sta bycad%+1
finens
byacad(i%)= bycad%
next i%
```

3.5. Envío de los datos requeridos por el oscilador programable OPIIC

En esta sección comprende la parte del programa que se encarga de recibir por puerto serie los datos que configuran a un oscilador programable, de modo que éste genere la señal de reloj requerida en un momento dado por el filtro de capacitancia conmutada presente en el sistema. El OPIIC tiene una interfaz IIC, empleando la propia presente en el MCU es como finalmente se envían los datos requeridos al OPIIC.

La parte central de este bloque del programa es el protocolo de comuni-

cación IIC, que será explicado a continuación.

3.5.1. Circuitos Inter - Integrados (IIC)

Este otro protocolo de comunicación serie de tipo síncrono fue creado por la empresa Philips Electronics, con la finalidad de crear una interfaz de comunicación de bajo costo.

Este protocolo trabaja bajo una arquitectura maestro - esclavo basada únicamente en dos líneas: La línea de datos seriales (SDA) y la línea correspondiente al reloj (SCL). La línea SCL es usada para sincronizar al dispositivo maestro con el esclavo, mientras que SDA es la encargada del intercambio de datos entre el maestro y el esclavo. El protocolo especifica que ambas líneas necesitan de resistencias de pull up colocadas externamente para tener un correcto funcionamiento.

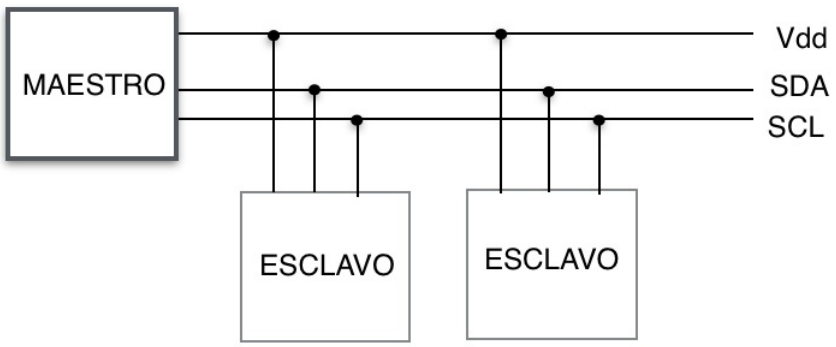

Figura 3.4: Estructura del protocolo IIC

La interfaz IIC es muy usada para tener comunicación con una amplia gama de circuitos integrados. La ventaja que presenta IIC es que se necesitan menos conexiones para lograr la comunicación, además de que pueden compartir el mismo bus hasta 127 dispositivos. En este proyecto de tesis, la interfaz IIC es usada para comunicarse con el OPIIC.

Las características del módulo IIC integrado se detallan en el capítulo 13 de la hoja de datos del MCU [1], aquí se presentan las más relevantes:

- Operación multi - maestro.
- Frecuencias de reloj programables.

- Interrupciones en la transferencia byte por byte.
- Generación y detección de señal de inicio y de paro.

Transmisión de datos

Cuando el bus de datos no está en uso, la línea SDA permanece en estado lógico alto y se puede iniciar la transmisión. El dispositivo maestro inicia el enlace al enviar una señal de inicio, definida como una transición de estado alto a bajo de la línea SDA, siempre y cuando el reloj haya sido configurado correctamente y esté activo. Esta transición indica el inicio de una nueva transmisión y pone activos a los esclavos.

Después de la señal de inicio inmediatamente se tienen que enviar siete bits, que son dirección del dispositivo esclavo a controlar, seguida por un bit de escritura/lectura. Este último bit decide la dirección de la información, es decir, si es 1 indica que el esclavo transmite los datos al maestro (el maestro lee), si es 0 indica que el flujo de datos es de maestro a esclavo (el maestro escribe).

Después de que el direccionamiento fue exitoso comienza la transmisión de la información byte por byte. Cada byte es transferido bit por bit en cada pulso de reloj. Después de que se envían todos los bytes necesarios, el maestro es el encargado de terminar la comunicación. Esto se logra generando una señal de paro en la línea SDA, la cual pone ésta en estado lógico alto.

3.5.2. Configuración del módulo IIC del MCU

La operación del módulo IIC está basada en cinco registros internos del MCU. El núcleo de operación del módulo IIC es el registro de control (IICC). Los otros cuatro registros que controlan el módulo son el registro de estado (IIC1S), el registro de datos (IIC1D), el registro de dirección (IIC1A) y el registro de divisor de frecuencia (IIC1F). En el capítulo 13 de la hoja de datos del MCU [1] se explica con detalle cada uno de estos registros y los bits de control correspondientes a cada uno de ellos.

Para podernos comunicar a través del MCU con dispositivos externos se requiere configurar el MCU. Para usar el puerto C como módulo IIC y no como un puerto de propósito general, se tiene que habilitar como módulo IIC y se hace a través del registro IICC.

Otro parámetro importante que se tiene que configurar es la velocidad de transmisión. Usualmente la velocidad establecida es de 100 kbits/s, aunque depende de los dispositivos a controlar. Para configurar el módulo IIC del MCU a esa velocidad se tiene que asignar el valor $56 hexadecimal al registro IIC1F.

3.5.3. Código del programa asociado al envío de datos al OPIIC

Una vez habilitado y configurado el módulo IIC, se encuentra listo para poder iniciar la comunicación. El diagrama de flujo de la figura 3.5 representa el proceso que se siguió para enviar los datos que configuran al oscilador programable. El diagrama muestra que primero se tiene que comprobar si el bus IIC está libre, si lo está se procede a configurar el MCU como el dispositivo maestro y se envía una señal de inicio de transmisión, además se configura la dirección de datos de maestro a esclavo.

De acuerdo con el protocolo de transmisión por IIC explicado anteriormente, después de la señal de inicio se envía la dirección del dispositivo esclavo a controlar y enseguida los bytes de datos. La dirección del oscilador programable es el valor hexadecimal $2e (información más detallada se puede encontrar en la hoja de datos del oscilador [4]). Los otros dos bytes de configuración del oscilador son recibidos por puerto serie a través de la subrutina lee#car y enviados por una subrutina que envía los datos por el bus IIC. El anterior proceso se lleva a cabo mediante las siguientes líneas de código:

```
lda #$2e
sta iic1d
jsr txiic

jsr lee#car
sta iic1d
jsr txiic

jsr lee#car
sta iic1d
jsr txiic
```

La subrutina *txiic* de las siguientes líneas de código espera a que finalice

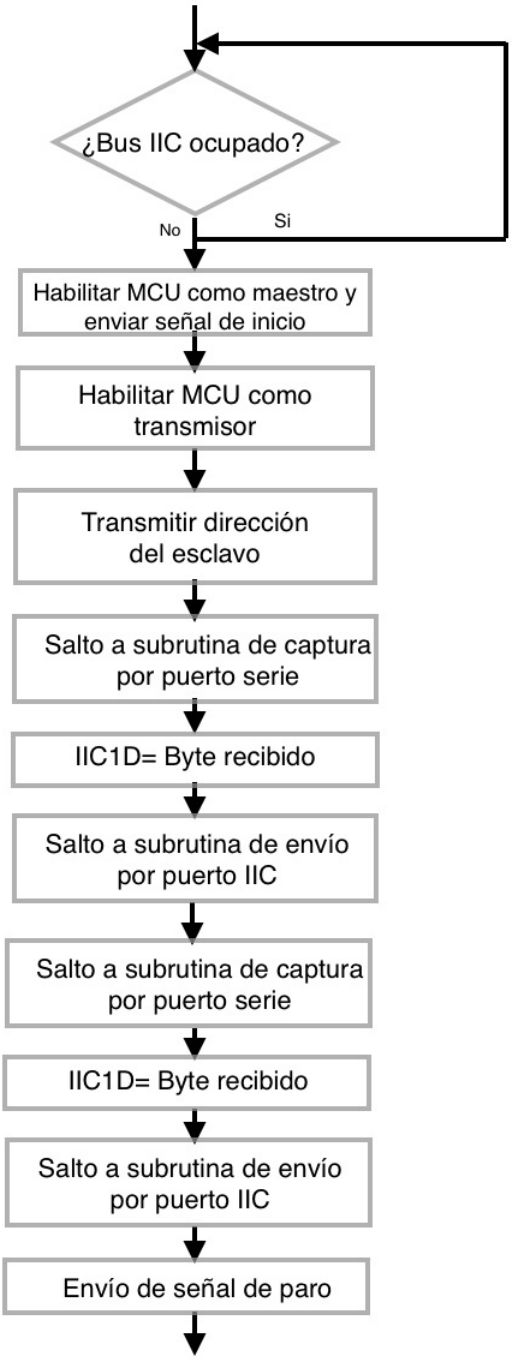

Figura 3.5: Diagrama de flujo para el proceso de envío de datos al OPIIC

la transmisión de un byte para que el programa pueda seguir su avance:

```
txiic: brclr 1,iic1s,txiic
bset 1,iic1s
brset 0,iic1s,fintx
fintx: rts
```

Para poner fin a la transmisión se envía la señal de paro, mediante la siguiente línea:

```
bclr 5,iic1c
```

3.6. Uso del evento de overflow del temporizador del MCU para la validación del periodo de muestreo

Para fines de la generación de una señal con un determinado periodo y número de muestras por periodo, se requerirá que la tabla asociada se recicle en el DAC con un periodo de muestreo (T_m), el cual estaría dado por la siguiente expresión:

$$T_m = \frac{T_p}{nmp} \tag{3.1}$$

Donde :

T_p es el periodo deseado para la señal a generar .

nmp es el número de muestras por periodo determinado por el usuario.

Para validar el T_m se configuró el temporizador 1 del MCU de modo que éste genere una interrupción por sobreflujo (overflow) a intervalos de tiempo iguales al valor de T_m requerido en un momento dado. En la subrutina de interrupción asociada se realiza el reciclamiento de la tabla en el DAC.

Los parámetros de T_p y nmp son requeridos al usuario por el software ejecutable en la PC.

3.6.1. Configuración del temporizador para fines del periodo de muestreo requerido.

Ls registros del MCU que controlan el funcionamiento del temporizador para fines del evento de overflow son TPM1SC, TPM1MODH y TPM1MODL.

Como es bien conocido para este MCU, para lograr un determinado valor de tiempo entre sobreflujos del temporizador, se requiere que el contador de éste lleve a cabo un determinado número de cuentas (NC), el cual está dado por la siguiente expresión:

$$NC = \frac{T_{ovf}}{T_b P_e} \tag{3.2}$$

Donde:

T_{ovf} es el valor del tiempo entre overflow requerido, que como se explicó anteriormente es igual al valor de T_m.

T_b es el periodo del reloj de bus al que opera el MCU. Para el MCU presente en la tarjeta MINICON_08GT este valor es 50 [ns].

P_e es el preescalamiento requerido en un momento dado. Su valor puede ser: 1, 2, 4, 8, 16, 32, 64 ó 128. Para fines de esta tesis, acorde con los parámetros asociados con la señal que se genere en un momento dado, el valor de P_e puede ser 1 ó 64.

Para lograr un determinado T_{ovf} es necesario que en los registros TPM1MODH y TPM1MODL se cargue el valor NC-1. En el registro TPM1SC se debe cargar lo propio de modo que el preesclamiento sea el requerido, el reloj primario del temporizador sea el reloj de bus, y que el bit de habilitación local de la interrupción de overflow (TOIE) esté en 1. Se sugiere ver la funcionalidad de este registro en el capítulo 10 de la hoja de datos del MCU [1].

Después de configurar y activar la interrupción mediante los registros TPM1SC, TPM1MODH y TPM1MODL, la rutina de interrupción se ejecutará con un tiempo entre ejecuciones igual a T_{ovf}, que es el periodo de muestreo requerido.

3.6.2. Rutina de interrupción

Aquí es donde se recicla el arreglo de bytes que contiene el valor de cada muestra. Antes, los valores del número de muestras por periodo ($nmp\,\%$) y de la dirección inicial de la tabla se deben guardar en variables auxiliares,

para que estas últimas sean las manipuladas en la rutina de interrupción.

Estas nuevas variables son:

nmpd %: variable de tipo *integer* que almacenará el número de muestras por periodo.

apuntab %: variable tipo *integer* a la cual se le cargará la dirección inicial de la tabla que contiene los valores de cada muestra.

Recordando que el arreglo *byacad*() contiene la tabla con los valores de cada muestra, la dirección en memoria donde inicia dicha tabla se guarda en la variable *apuntab* %. El número de muestras por periodo contenido en *nmp* % se guarda en *nmpd* %. Lo anterior se logra con las siguientes líneas de código:

```
ldhx #byacad
sthx apuntab %
ldhx nmp %
sthx nmpd %
```

Teniendo los datos en las nuevas variables, se procede a explicar la rutina de interrupción, cuyo funcionamiento queda representado por el diagrama de flujo de la figura 3.6.

La rutina de interrupción llamada *servovf* comienza por limpiar la bandera de overflow, la cual se activa al producirse un sobreflujo en el temporizador. Esto se tiene que hacer puesto que si la bandera permanece activa no se puede ingresar nuevamente a la rutina de interrupción hasta que ésta se desactive. En la hoja de datos del MCU [1] se indica que para regresar a cero esta bandera, se debe hacer una lectura del registro TPM1SC seguida por una puesta a cero del bit propio de la bandera, esto podría lograrse entre otras formas mediante el siguiente tramo de código en ensamblador:

```
lda tpm1sc@
bclr 7,tpm1sc@
```

Posteriormente se carga la dirección en memoria de la muestra presente que se va a colocar en el puerto A, que está contenida en la variable apuntab %:

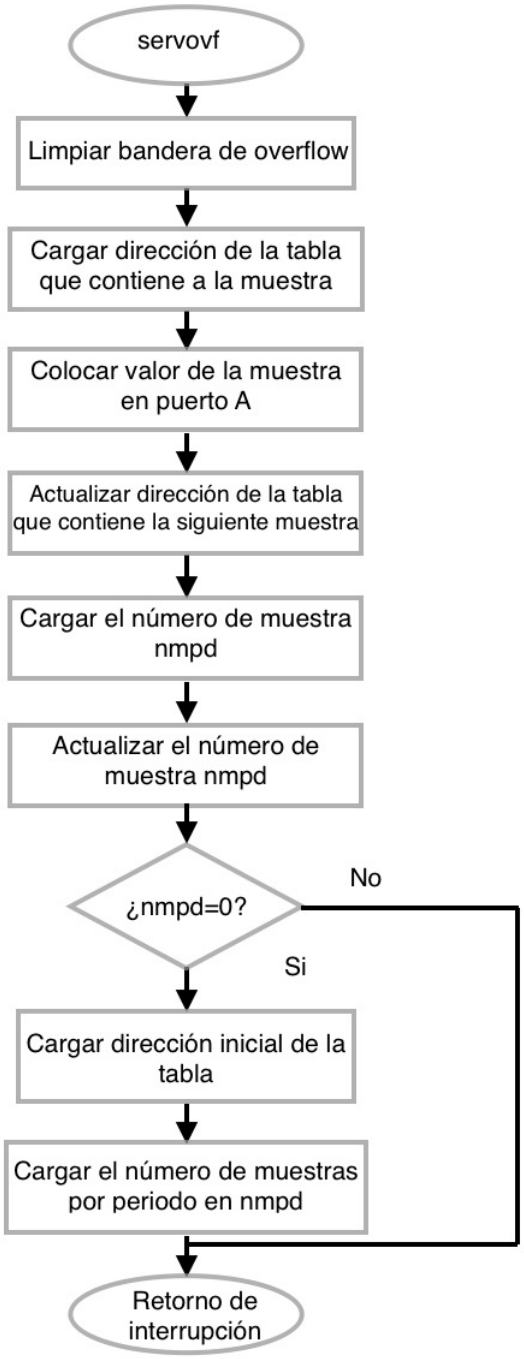

Figura 3.6: Diagrama de flujo de la rutina de interrupción por overflow del temporizador.

ldhx apuntab%

Mediante las siguientes líneas se coloca la muestra en el puerto A:

lda $01,x
sta ptad@

A continuación se actualiza la dirección de memoria, que ahora corresponderá a la dirección de la siguiente muestra a colocar:

aix #$02
sthx apuntab%

El siguiente bloque es el bloque de la carga del número de muestra que se colocó, y se actualiza al número de muestras faltantes por colocar:

ldhx nmpd%
aix #$ff
sthx nmpd%

Si aún faltan muestras por colocar, se da un salto a retorno de interrupción, conservando los valores actuales de *nmpd*% y *apuntab*%, de modo que la siguiente vez que entre a la rutina se tengan los valores que corresponden a la siguiente muestra a colocar.

Si ya se colocaron todas las muestras de un periodo de la señal, las variables *apuntab*% y *nmpd*% se cargan con los valores iniciales, es decir, *apuntab*% se carga con la dirección inicial de memoria donde inicia la tabla y *nmpd*% se carga con el número de muestras por periodo totales. De esta forma se logra el reciclado del arreglo de bytes almacenado en *byacad*():

ldhx #byacad
sthx apuntab%
ldhx nmp%
sthx nmpd%

3.6.3. Restricciones para el periodo de muestreo T_m

El periodo de muestro T_m está limitado por dos factores: el periodo del reloj de bus al que opera el MCU y del tiempo asociado con la ejecución de

las instrucciones utilizadas en la rutina de interrupción. Cada instrucción se lleva a cabo en un determinado número de ciclos de bus del reloj. En el capítulo 8 de la hoja de datos del MCU [1] se presenta una tabla con todas las instrucciones y los ciclos de reloj que requiere cada una de ellas dependiendo del modo de direccionamiento que se utilice.

Si se analiza el código de la rutina de interrupción y se toman en cuenta otros factores como el número de ciclos que toma el salto a la misma, se tiene que la rutina de interrupción requiere de 69 ciclos de reloj, que multiplicado por el periodo de bus (50 [ns]) es igual a 3.45[μs]. Este es el periodo mínimo que podría existir ente cada muestra, sin embargo para este trabajo el periodo mínimo se limitará a 5 [μs] puesto que al momento de desplegar la señal se podrán efectuar acciones como el cambio de nivel de offset, de amplitud y restablecimiento del MCU, que requieren de algunas instrucciones con su respectivo número de ciclos de reloj.

3.7. Envío de los datos requeridos por los potenciómetros digitales POTSPI

Este es el último bloque del programa, en el cual se reciben por puerto serie los datos de configuración del POTSPI, cuya función es configurar los valores de resistencia requeridos para ajustar los parámetros de amplitud y nivel de offset de la señal, definidos previamente por el usuario en el software que corre en la PC. Gracias a la estructura del programa, tanto la amplitud de la señal como el nivel de offset pueden ser ajustados en tiempo real a través del programa de la PC. El POTSPI trabaja con una interfaz SPI, por lo que se usa la propia del MCU para el envío de datos al POTSPI. Para tener noción del funcionamiento del protocolo SPI, éste se explica a continuación.

3.7.1. SPI Módulo de interfaz de periféricos serie

El protocolo SPI fue diseñado por Motorola, con el objetivo de crear un estándar de comunicaciones que permita una fácil interconexión entre los microprocesadores y los circuitos integrados como convertidores analógico-digitales, memorias, o en el caso del proyecto a realizar en esta tesis, un potenciómetro digital.

El MC9S08GT60 incluye un módulo SPI explicado a detalle en el capítulo 12 de la hoja de datos del MCU[1], sus principales características se presentan a continuación:

- Capacidad para operar como maestro o esclavo.
- Transmisión y recepción de datos al mismo tiempo (full duplex).
- Opción de usar sólo un cable para un intercambio de información bidireccional.
- Tasa de bits (bit rate) programable.
- Operación con interrupciones por diferentes eventos del SPI.

Este protocolo de comunicación síncrona utiliza una arquitectura maestro-esclavo, y como se aprecia en la figura 3.7 está basado en cuatro líneas:

SCLK (CLOCK): Marca la sincronización, en cada pulso del reloj se envía o recibe un bit.

MOSI (Master Output Slave Input): Los datos salen del dispositivo configurado como maestro y entran al dispositivo esclavo.

MISO (Master Input Slave Output): Los datos salen del dispositivo esclavo y entran al maestro.

SS (Slave Select): Mediante esta señal lógica el maestro elige o activa un esclavo.

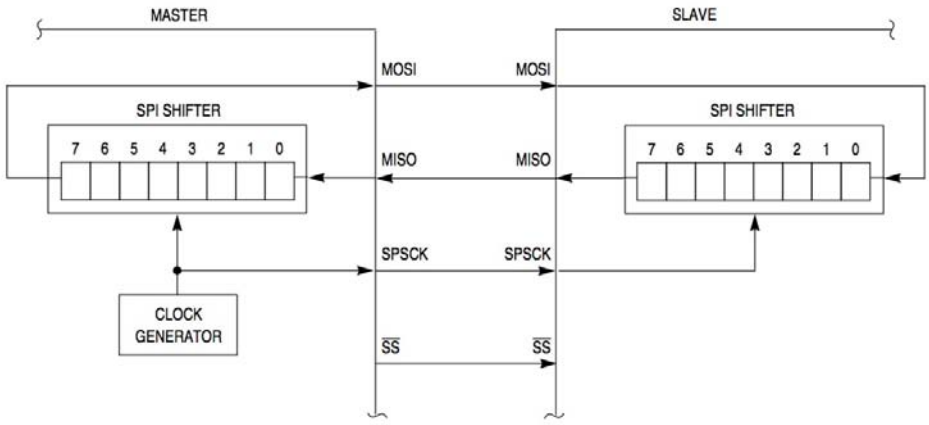

Figura 3.7: Protocolo de comunicacón SPI.

Transmisión de datos

Para iniciar la comunicación, el dispositivo maestro configura la señal de reloj (SCLK) usando una frecuencia soportada por el dispositivo esclavo. Mediante la línea SS, el maestro selecciona y activa al esclavo llevando esta línea a un nivel lógico igual a cero. Aquí es donde empieza la transmisión, por cada pulso del reloj el maestro estará enviando un bit al mismo tiempo que está siendo leído por el esclavo.

El intercambio de información a través de este protocolo en este microcontrolador se hace por palabras de ocho bits. Normalmente los datos son desplazados primero por el bit más significativo, cuando se terminan de transmitir los ocho bits el proceso puede repetirse para enviar datos adicionales. Cuando se terminan de enviar los datos el dispositivo maestro para la señal de reloj y lleva la línea SS a un nivel lógico igual a uno, lo que pone fin a la transmisión.

3.7.2. Configuración del módulo SPI del MCU

El control del módulo SPI interno del MCU se logra con los siguientes registros: dos registros de control (SPIC1 y SPIC2), el registro de estado (SPIS), registro de baud rate (SPIBR) y el registro de datos (SPID).

Se debe cargar lo propio en SPIC1 para habilitar el módulo SPI del MCU, así como para configurar al MCU como dispositivo maestro en la transmisión. La velocidad de la transmisión corresponde al valor que se le asigna a SPIBR.

La descripción detallada de los bits que conforman cada registro del periférico SPI se puede consultar en el capítulo 12 de la hoja de datos del MCU [1].

3.7.3. Interfazado de los potenciómetros digitales y restablecimiento del MCU desde la PC anfitriona

El último tramo de código del programa está asociado con el interfazado entre los potenciómetros digitales y la PC, de modo que éstos se ajusten a los valores propios requeridos por determinados valores de amplitud y offset de la señal generada, especificados por el usuario en la computadora anfitriona. Otro aspecto realizado en este tramo de código es el asociado con el restablecimiento del MCU desde la PC anfitriona, lo cual se requiere efectuar cuando se desea cambiar la señal desplegada en cuanto a forma y frecuencia.

A continuación se describe este tramo de código.

El diagrama de flujo de la figura 3.8 representa el proceso que se lleva a cabo en el último bloque del programa.

Mientras la señal es desplegada en el puerto A por acción de una interrupción por overflow del temporizador, el programa se queda a la espera de un byte que es recibido por puerto serie, el cual determinará la siguiente acción a ejecutar. El valor del byte que se recibe depende de la acción que se quiera ejecutar en el programa de la PC, que puede ser un RESET al MCU o modificar la amplitud y el nivel de offset de la señal.

Si el byte recibido es igual a 1, la acción a ejecutarse es detener la interrupción y limpiar el puerto A para que deje de desplegarse la señal. Por último, se da un salto a la dirección de memoria $fe7d, esta dirección se encuentra definida en el software base del dispositivo CHIPBAS8GT para que se invoque el receptor de comandos. De esta manera es como se finaliza la ejecución del programa y el MCU queda en modo receptor de comandos. La anterior acción se realiza con el siguiente tramo de código:

```
bclr 6,tpm1sc@
mov #$00,ptad@
jmp $fe7d
```

En cambio, si el byte recibido es igual a 0, la acción siguiente es enviar los tres bytes de configuración que necesitan los potenciómetros digitales. Cada byte es recibido por el puerto serie y enviado al POTSPI por el módulo SPI interno del MCU:

```
jsr lee#car
sta spid@
jsr txspi

jsr lee#car
sta spid@
jsr txspi

jsr lee#car
sta spid@
jsr txspi
```

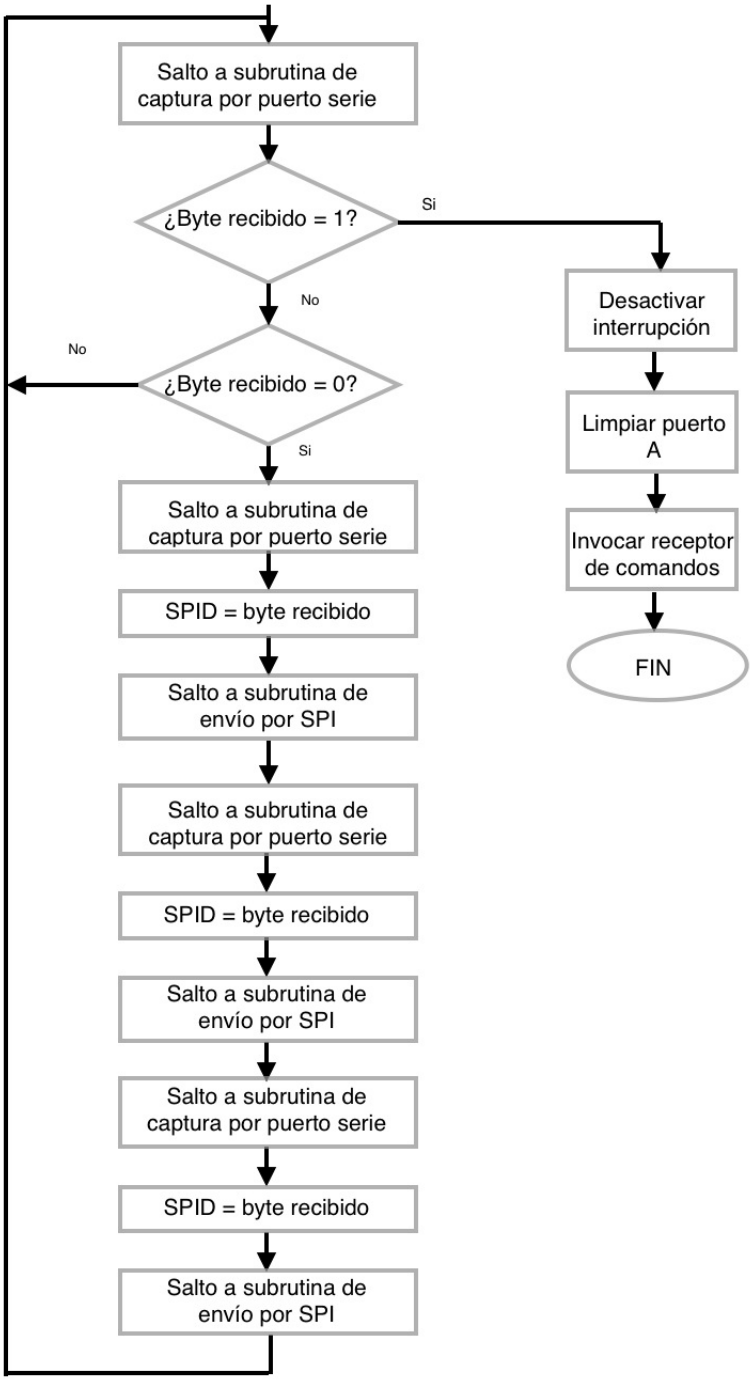

Figura 3.8: Diagrama de flujo para el proceso de transmisión de datos al POTSPI y reset del microcontrolador .

La subrutina llamada *txspi* espera a que el buffer de transmisión esté vacío, es decir, que se haya completado la transmisión de un byte.

Hasta aquí se explicaron los elementos esenciales que conforman al software ejecutable en el MCU, el código completo puede verse en la sección de anexos.

4 | Desarrollo de la interfaz de usuario ejecutable en la computadora PC

En los anteriores capítulos se ha mencionado que la forma en que el usuario define los parámetros básicos de la señal a generar es mediante un software ejecutable en la PC. Es por eso que es necesaria una interfaz humano-máquina que sea intuitiva y amigable para el usuario, en la que pueda definir la forma de la señal y parámetros esenciales como el periodo y el número de muestras que la conforman. Dicho software funciona como un agente de comunicación entre la computadora y el dispositivo CHIBAS8GT.

En el presente capítulo se llevará a cabo el análisis del software desarrollado denominado "Software manejador del GSPFP", ejecutable en el sistema operativo WINDOWS. Dicho software fue desarrollado en Visual Basic, ya que nos provee de herramientas y controles que facilitan la creación de una interfaz de usuario.

4.1. Características de la interfaz de usuario (IU)

Para fines de la generación de una señal periódica de forma predeterminada, la IU deberá tener las siguientes facilidades:

- Se podrá definir la forma de onda del periodo básico.
- Definir el periodo de la señal a generar.
- Definir el número de muestras por periodo.
- Guardar en disco cualquier forma de onda determinada por el usuario.

- Recuperar de disco una señal para su reproducción.
- Establecer y cambiar en tiempo real la amplitud de la señal generada.
- Establecer y cambiar en tiempo real el nivel de offset de la señal generada.
- Descargar la señal a la tarjeta MINICON_08GT.

4.2. Elementos y controles que conforman a la IU

A continuación se explicarán los elementos básicos de la IU, se explicará el funcionamiento de los controles presentes así como algunos cálculos que se llevan a cabo en el programa. En la figura 6.1 se puede observar el aspecto que tiene la IU en la que se aprecian los elementos que la conforman.

4.2.1. El control de puerto serie

Visual Basic implementa un control para la comunicación por puerto serie con otros dispositivos externos a la computadora. Este control llamado *MSComm* permite la transmisión y recepción de un byte por puerto serie. Gracias a este control se puede configurar la velocidad de transmisión, formato de transmisión y el número de puerto COM al que se encuentra conectado el dispositivo con el cual se establecerá la comunicación. Este control resulta esencial para la realización del proyecto de tesis, pues es usado para que a través de él se envíen los datos correspondientes al MCU encargado de desplegar la señal.

4.2.2. Definición de la forma de onda del periodo básico en una pantalla virtual

Para definir la forma de onda se diseñó una pantalla virtual (figura 4.1) que se asemeja a la pantalla de un osciloscopio. Para ello se utilizó el control *PictureBox*. Las líneas verticales y horizontales que forman la cuadrícula de la pantalla virtual se dibujaron con el control *line*.

Para tener un control sobre el posicionamiento de cada punto se tiene un sistema de coordenadas X-Y, cuyo origen se colocó en la esquina inferior

Figura 4.1: Pantalla virtual donde se define la forma de onda

izquierda de la pantalla virtual.

El dibujado de la señal se hace mediante tramos rectos, que se forman al unir los puntos que el usuario coloca al hacer click sobre el lugar deseado de la pantalla virtual. Si se intenta colocar un punto cuyo valor sobre el eje X sea igual o menor al punto anterior, el programa desplegará un cuadro de diálogo como el de la figura 4.2.

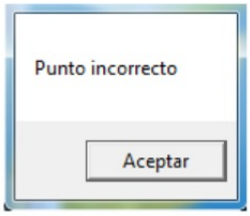

Figura 4.2: Cuadro de diálogo que se presenta al colocar incorrectamente un punto.

4.2.3. Controles para la edición de la señal

Se tienen dos botones que dan facilidades para modificar una señal que se está dibujando en un momento dado (figura 4.3). Al presionar el botón $< Borrar >$ se borra completamente la señal dibujada en la pantalla virtual, el botón *<Borrar última línea>* se activa cuando se traza la primera línea y se desactiva cuando no hay ninguna línea, y como su nombre lo indica sirve para borrar el último tramo de recta dibujado.

Figura 4.3: Botones de la IU para la edición de la forma de la señal en la pantalla virtual.

4.2.4. Definición del número de muestras por periodo y el periodo de la señal

Para definir los parámetros de número de muestras por periodo y periodo de la señal a generar se implementaron cuadros de texto donde el usuario puede escribir los valores deseados (figura 4.4).

El número de muestras por periodo está limitado a 512 muestras como máximo, que son las suficientes para generar cualquier señal de forma bien definida y 50 muestras como mínimo, pues con un número menor de muestras la forma de la señal generada no se define correctamente y por tanto no es reconocible como la forma de onda deseada en un momento dado.

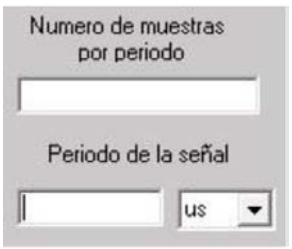

Figura 4.4: Elementos de la IU para definir el periodo y número de muestras

El periodo de la señal se especifica en el cuadro de texto inferior, dicho periodo puede especificarse en microsegundos, milisegundos o segundos. Para

el cambio en las unidades de tiempo se colocó un control que despliega una lista con las tres unidades de tiempo para seleccionar.

4.2.5. Control de la amplitud y nivel de offset de la señal

Se implementó el control de los parámetros de amplitud y nivel de offset mediante un control *scrollbar* (figura 4.5). El valor del *scrollbar* que cambia la amplitud de la señal va de 0 a 5 [Vp], mientras que el valor del nivel de offset va de -5 a 5 [V]. Es importante mencionar que estos parámetros pueden ser modificados en tiempo real, es decir, se pueden cambiar los valores mientras la señal está siendo desplegada por el MCU.

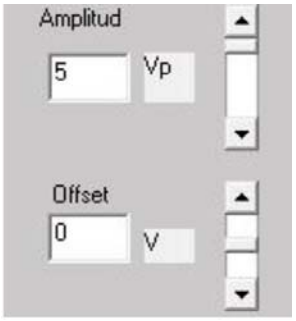

Figura 4.5: Elementos de la IU para el control de amplitud y offset

Cada que se modifica un valor, se envían por puerto serie al MCU el valor de los bytes que son necesarios para configurar el POTSPI, explicado en el capítulo anterior. La determinación del valor de cada byte a enviar para el ajuste del POTSPI será explicada en el capítulo 5, pues es necesario el análisis de los circuitos diseñados para la aplicación.

4.2.6. Descarga de los datos de la señal al MCU

El botón *Bajar señal* (figura 4.6) se debe presionar cuando se quieren transmitir los datos de la señal al MCU. Previamente, se tuvo que haber definido la forma de onda, el número de muestras por periodo y el periodo de la señal. También se pudieron definir la amplitud y el nivel de offset, sin embargo estos parámetros se encuentran por defecto en 5[Vp] y y 0[V] respectivamente.

Figura 4.6: Botón para la descarga de datos al MCU.

Al presionar el botón mencionado, se procede a hacer una serie de cálculos para determinar los valores que serán enviados al MCU.

Primeramente se determina si es posible generar la señal con los parámetros establecidos mediante la siguiente ecuación:

$$T_m = \frac{T \quad mult}{nmp} \tag{4.1}$$

Donde:

T_m es el periodo entre muestras.

T es el periodo de la señal establecido por el usuario.

nmp es el número de muestras por periodo.

$mult$ es un multiplicador cuyo valor depende de la unidad de tiempo seleccionada en el control combo cuando se define el periodo de la señal.

El valor de T_m depende del valor de nmp usado en un momento dado. Por las restricciones explicadas en la sección 3.6.3, el valor mínimo de T_m es de 5 microsegundos. Por ejemplo, si se quiere generar una señal con 200 muestras por periodo, el valor mínimo del periodo de la señal es de 1 milisegundo, o bien, la frecuencia máxima de la señal para ese valor de nmp es de 1 [kHz].

Si el valor de T_m calculado es menor a 5 microsegundos, el programa desplegará el mensaje de la figura 4.7. Se tendrá que colocar un valor de nmp y T de tal forma que la relación entre ellos de un valor de T_m mayor o igual a 5 microsegundos para que la señal pueda generarse.

Si con los parámetros definidos por el usuario es posible desplegar la señal, se continúa con el cálculo de los valores que serán enviados al MCU.

A continuación se calcula el valor de NC necesario para la configuración del temporizador del MCU, explicado en el capítulo 3.

$$NC = \frac{T_m}{50[ns] \quad P_e} \tag{4.2}$$

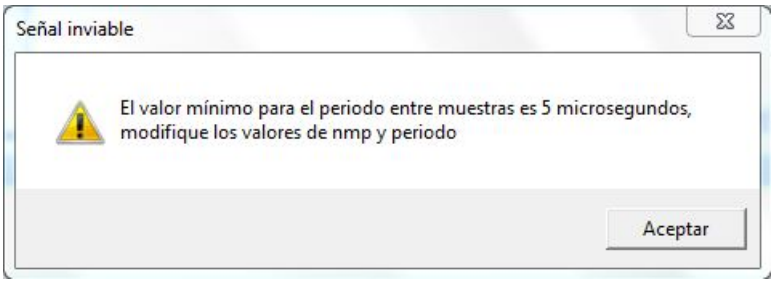

Figura 4.7: Cuadro de diálogo cuando se presenta una señal inviable.

Recordando que NC es el valor del número de cuentas que el temporizador debe realizar para obtener el T_m calculado. El valor de P_e es el valor de preescalamiento necesario para que el MCU genere el desbordamiento del overflow con el T_m deseado. Este último puede valer 1 ó 64, si el valor de T_m está comprendido entre 5[μs] y 1 [ms] se usará $P_e = 1$. Si el valor de T_m es mayor a 1 [ms] el valor de $P_e = 64$.

4.2.7. Cálculo de los coeficientes de las rectas que conforman la señal

Se explicó anteriormente que en la pantalla virtual de la figura 4.1 se dibuja la señal mediante tramos rectos que son definidos por puntos que el usuario define sobre la pantalla.

También se mencionó que se creó un sistema de coordenadas X-Y en la pantalla virtual, por lo que cada recta quedaría definida como en la figura 4.8. Cada punto definido por el usuario tendrá una coordenada (x_n, y_n).

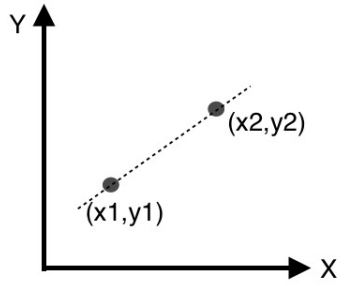

Figura 4.8: Representación de los tramos de recta al unir dos puntos en la pantalla virtual.

Teniendo los dos puntos que forman un tramo recto de la señal, se puede calcular la función lineal que representa a cada recta. La ecuación de cada tramo de recta está dada por:

$$y = mx + b \tag{4.3}$$

Primero se halla la pendiente m del tramo recto, con la siguiente expresión:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{4.4}$$

Después encontramos el valor de b, reemplazando en la ecuación 4.3 el valor de m obtenido y sustituyendo los valores de x y de y por las coordenadas de uno de los puntos.

4.2.8. Cálculo del valor de cada muestra en el eje X

Una vez teniendo el valor de nmp y de las coordenadas del primer y último punto que forman a la señal, se determina el espacio entre muestras que habrá en el eje X mediante la siguiente ecuación:

$$E_m = \frac{x_n - x_0}{nmp} \tag{4.5}$$

Donde:

E_m es el espacio entre muestras sobre el eje X.

x_n es el valor de la coordenada X del último punto de la señal.

x_0 es el valor de la coordenada X del primer punto de la señal.

De esta forma se obtiene el valor de la coordenada X de cada muestra, pues el valor en X de la primera muestra corresponde el valor de x_0, el valor de X la siguiente muestra sería $x_0 + E_m$ y así sucesivamente para las muestras faltantes.

4.2.9. Cálculo del valor de cada muestra en el eje Y

Ahora que ya se tiene la ecuación lineal de cada recta que conforma a la señal dibujada y el espacio entre muestras en el eje X, se calcula el valor de cada muestra en el eje Y. Para ello se hace un barrido de la señal en el eje X, sustituyendo el valor de la coordenada X de cada muestra en la ecuación lineal que le corresponde a cada tramo de recta.

4.2.10. Cálculo del valor del byte de cada muestra

Ahora ya se cuenta con el valor en el eje Y de cada muestra. Sin embargo, dichos valores se encuentran definidos por las unidades y las dimensiones que se manejan en la pantalla virtual. Dichas dimensiones se encuentran comprendidas entre 5500 y 4400 *twips* para el eje X y Y respectivamente. *Twips* son las unidades con la que se manejan los gráficos en Visual Basic.

La señal se desea desplegar en el puerto A del MCU, que es de 8 bits, por lo que el valor de cada muestra debe estar comprendido entre 0 y 255. De esta forma se logrará que a la salida del DAC la señal esté homologada a +10 y -10 [V].Con la siguiente ecuación se calcula el valor de cada byte a colocar en el DAC:

$$Bydac = 255\frac{y_m - y_{min}}{y_{max} - y_{min}} \tag{4.6}$$

Donde:

$Bydac$ es el valor final del byte de la muestra.

y_m es el valor en el eje Y de la muestra, en twips.

y_{min} es el valor mínimo en el eje Y de todas las muestras, en twips.

y_{max} es el valor máximo en eje Y de todas las muestras, en twips.

Por lo que se observa que en la ecuación 4.6, si y_m es igual a y_{min} el valor de Bydac es 0. Si y_m es igual a y_{max}, el valor de Bydac es 255. El valor de Bydac de las demás muestras quedará comprendido entre 0 y 255.

4.2.11. Transmisión de los datos al MCU

Una vez que ya se cuenta con todos los parámetros que van a generar la señal, ahora se tienen que enviar al MCU para que haga la función de desplegarla. Para ello se utilizó el control *MSComm* mencionado anteriormente y se creó una función de transmisión.

Los primeros bytes que se tienen que transmitir, corresponden a un proceso para poder ejecutar el programa en el MCU.

El programa que corre el MCU está guardado en su memoria, específicamente en la memoria FLASH. Al energizar la MINICON_08GT, ésta estará

en modo recepción de comandos. Para que ocurra un salto a ejecutar un programa que se encuentre cargado en una dirección determinada, es necesaria una cadena de bytes que debe recibir el MCU. Esta cadena consta de ocho bytes, que son los siguientes:

byte 0 = $0e
byte 1 = $01
byte 2 = $00
byte 3 = $01
byte 4 = $02
byte 5 = $cc
byte 6 = byte alto de la dirección inicial del programa a ejecutar.
byte 7 = dirección final del programa a ejecutar.

La dirección de la memoria FLASH corresponde a $182c, por lo que el byte 6 será igual a $18 y el byte 7 igual a $2c. La cadena de 8 bytes será enviada al MCU a través del puerto serie, cada vez que se requiera descargar los datos de una señal al MCU. Una vez recibida esta cadena, se ejecutará en el MCU el programa que corre en él y que se describió en el capítulo anterior.

Por último, usando la misma función de transmisión se envían los valores de los bytes que el MCU espera, con el siguiente orden: byte alto de *nmp*, byte bajo de *nmp*, byte alto de *xr*, byte bajo de *xr*, valor de cada muestra contenida en *Bydac*, byte alto que configura al OPIIC, byte bajo que configura al OPIIC, byte con el valor de NC y por último los tres bytes de configuración del POTSPI.

4.2.12. Restablecimiento del MCU desde el software que se ejecuta en la PC

Cuando una señal sea descargada y desplegada correctamente por el MCU, el botón de <RESET>(figura 4.9) brinda la posibilidad de eliminar la señal presente en el DAC y poner al MCU en modo recepción de comandos, para la posible recepción de datos de otra señal.

4.2.13. Facilidades para el manejo de archivos

El software se diseñó con las facilidades de un programa convencional: Se puede guardar una señal en un archivo con extensión *.sgn* y después ser recuperada para su reproducción. Las opciones que se despliegan al presionar

Figura 4.9: Botón presente en la IU para el restablecimiento del MCU.

en la barra de menús el botón *Archivo* se ven en la IU como en la figura 4.10 y son las sguientes:

- Nuevo
- Abrir
- Guardar
- Guardar como
- Salir

Figura 4.10: Facilidades de la IU para el manejo de archivos.

Los archivos *.sgn* son archivos de texto que contienen el número de puntos y las coordenadas de cada punto que conforman la señal dibujada en la pantalla virtual, al abrir un archivo de este tipo el programa leerá cada dato y dibujará la señal tal cual se guardó en el archivo.

Hasta aquí se explicaron las funcionalidades básicas del software ejecutable en la PC. El código fuente del programa desarrollado se encuentra en la sección de anexos.

5 | Diseño del hardware analógico y digital requerido

En este capítulo se mostrará el diseño del hardware analógico que conforma a la etapa de adecuación analógica, así como el funcionamiento hardware digital empleado para lograr obtener una señal con las especificaciones que se hayan definido previamente mediante la interfaz de usuario.

5.1. El convertidor digital - analógico

En los anteriores capítulos se explicó el software encargado de realizar la tarea de desplegar la señal en el puerto A de ocho bits del MCU, sin embargo, se necesita un convertidor digital - analógico (DAC) para transformar los datos binarios en la señal analógica deseada.

Para el diseño y pruebas del dispositivo objeto de esta tesis, se utilizó el circuito integrado DAC0800. Este circuito integrado es un convertidor digital - analógico de 8 bits de bajo costo y de alta velocidad de conversión. El DAC0800 implementa internamente una red R-2R o también llamada de escalera (figura 5.1).

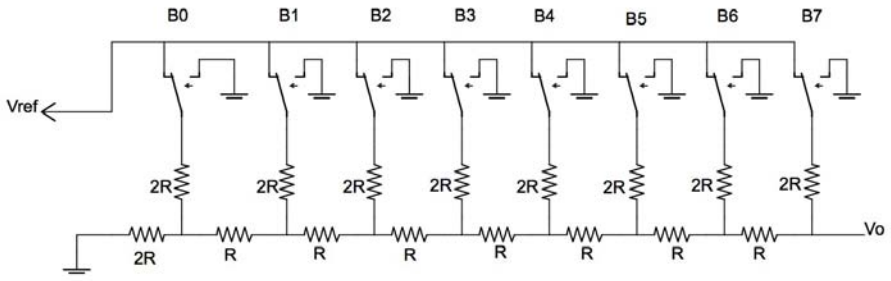

Figura 5.1: Red R-2R de 8 bits

La ventaja que presenta esta configuración es que sólo se necesitan dos valores de resistencias (R y 2R) y se puede construir un DAC con el número de bits deseado simplemente añadiendo más ramas R2R. En el caso del DAC0800, tiene 8 ramas R-2R al ser de 8 bits.

5.1.1. La red resistiva R-2R

En la figura 5.1 se observa que los 8 bits de datos binarios ingresan paralelamente a través de 8 líneas (B0 a B7), los cuales serán convertidos a un voltaje analógico equivalente (v_0) a través de la red resistiva R-2R.

Si se hace un análisis evaluando los equivalentes de Thevenin desde cualquiera de los nodos entre las resistencias, la resistencia equivalente de cualquiera de estos puntos será igual a R, independientemente del tamaño (número de bits) de la red.

Para calcular el voltaje a la salida (v_0) se usa el mismo análisis de los equivalentes de Thevenin y análisis por superposición de la red de la figura 5.1. El método de superposición indica que si se calcula de forma individual la contribución a v_0 de cada fuente de voltaje (con las demaś fuentes de voltaje en circuito corto y fuentes de corriente en circuito abierto), se puede sumar el resultado de la contribución de cada fuente para obtener el valor final de la salida v_0.

Tomando en cuenta lo anterior, el valor de v_0 estará dado por:

$$v_0 = v_{ref}\left(\frac{B0}{256} + \frac{B1}{128} + \frac{B2}{64} + \frac{B3}{32} + \frac{B4}{16} + \frac{B5}{8} + \frac{B6}{4} + \frac{B7}{2}\right) \tag{5.1}$$

Se puede observar que el bit más significativo es el que tiene mayor contribución, mientras que el bit menos significativo es el que tiene menor contribución.

5.1.2. Diseño del circuito del DAC

Se espera que la señal a la salida del DAC esté homologada a +10[V] y -10[V] y se sabe que la señal será desplegada por el puerto digital de 8 bits puede tomar un valor de 0 a 256, o bien, de 0x00 a 0xFF en notación hexadecimal. Por lo tanto, la conversión de la señal quedará definida de la siguiente manera:

$$0x00 = \text{-}10\ [V]$$
$$0xFF = +10[V]$$

Para lograr obtener esos valores de conversión es necesario utilizar el DAC0800 en su modalidad simétrica y un amplificador operacional, como se muestra en la figura 5.2. Utilizando esta configuración el voltaje de salida queda definido con la siguiente ecuación:

$$E_0 = \frac{-255}{256} + \frac{2X}{256} \tag{5.2}$$

Donde:

E_0 es el voltaje a la salida del amplificador operacional

X es el número binario desplegado en el puerto A del MCU, transformado a base decimal.

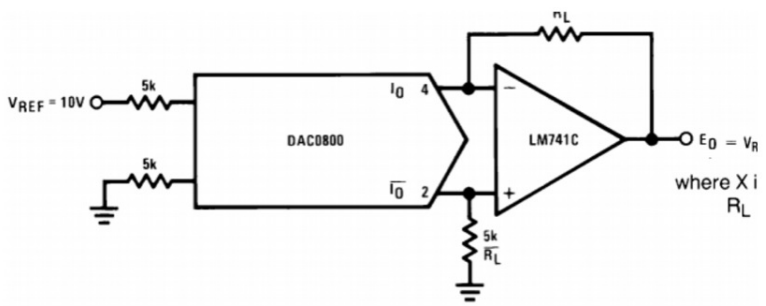

Figura 5.2: Configuración simétrica del DAC0800

El diseño del circuito convertidor analógico - digital se presenta en la figura 5.3. Se observa que en el diseño se utilizó la configuración simétrica del DAC0800 mostrada en la figura 5.2. Los datos binarios contenidos en los 8 bits que son desplegados por el MCU sirven como entrada a los pines correspondientes en el DAC.

Para el fin deseado, la referencia del DAC0800 (v_{ref}) debe ser igual a 10 [V], la cual se logra ajustar con el potenciómetro P1.

A la salida del amplificador operacional, se tendrá la señal analógica homologada a +10 [V] y -10 [V], que posteriormente será adecuada para obtener la señal deseada por el usuario.

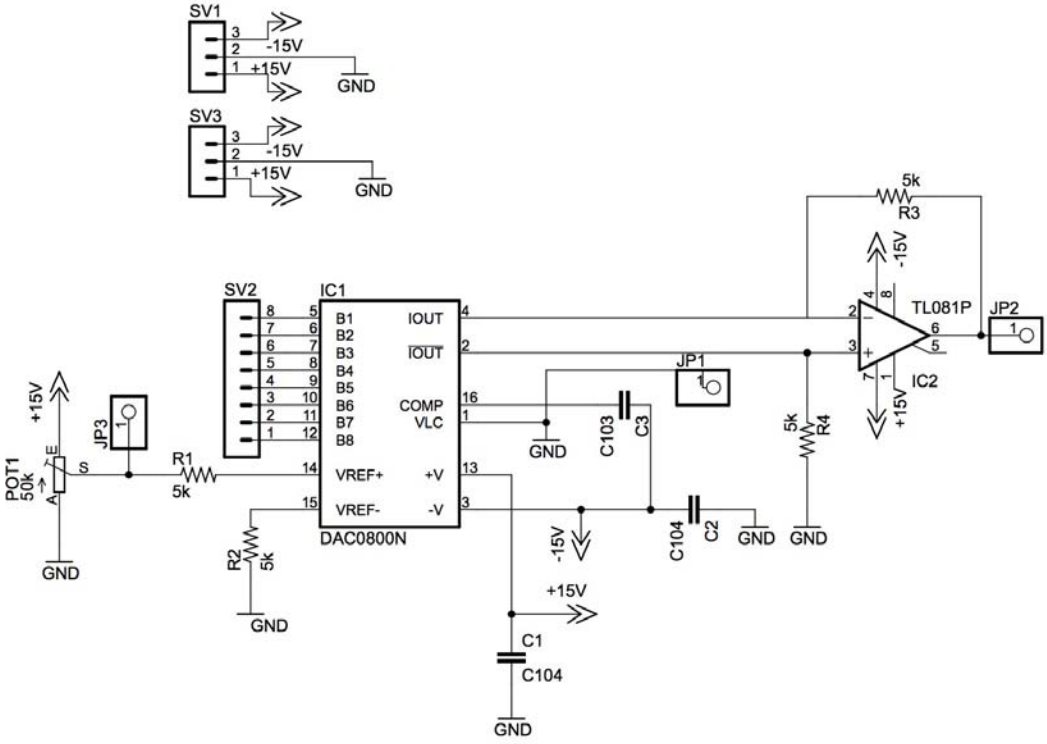

Figura 5.3: Circuito convertidor digital - analógico

5.2. Atenuador de entrada del filtro

La señal a la salida del DAC está ajustada para que sea de 20 [Vpp], pero por las características eléctricas del filtro LMF100 se tiene que adecuar la señal a un nivel (máximo 5 [V_{pp}], en este diseño se ajustó a 4 [V_{pp}]) . Para lograr este fin se colocó un amplificador operacional en configuración inversora (figura 5.4) cuya ganancia está dada por la siguiente ecuación:

$$V_{out} = -\frac{R2}{R1}V_{in} \tag{5.3}$$

Si se elige R1 = 20 kΩ y R2 = 100 kΩ, la ganancia del amplificador será -0.2, por lo que la señal a la salida del amplificador quedará invertida y reducida a

4 [V_{pp}], adecuada para un funcionamiento óptimo del filtro. Como se aprecia en la figura 5.4, esta etapa se coloca a la salida del DAC y funciona como entrada al filtro.

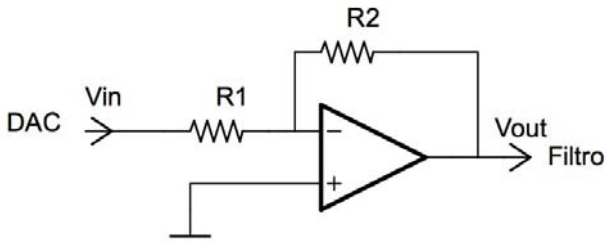

Figura 5.4: Circuito atenuador de entrada del filtro.

5.3. Filtrado de la señal

En el capítulo anterior se vio la relación entre el número de muestras y el periodo de la señal, de donde se sabe que a menor número de muestras por periodo se puede conseguir una mayor frecuencia. No obstante, si se desea generar una señal con un número de muestras pequeño la señal no quedará bien definida, pues se le sumarán componentes de alta frecuencia perceptibles entre cada muestra. Para eliminar estos componentes la señal debe pasar por una etapa de filtrado.

Un filtro es un circuito electrónico cuya función es dejar pasar una señal que se encuentra en una banda de frecuencias especificada, mientras que atenúa todas las señales que están fuera de esta banda. Existen diferentes tipos de filtros, el que es útil para fines de la aplicación es un filtro de tipo paso bajo.

Un filtro paso bajo permite sólo el paso de frecuencias por debajo de una frecuencia llamada frecuencia de corte (f_c). En otras palabras, con este filtro se obtendrá una salida de voltaje constante hasta la frecuencia de corte, conforme la frecuencia aumenta arriba de f_c, el voltaje de salida se atenúa. En la figura 5.5 se ilustra la respuesta en frecuencia de un filtro paso bajo.

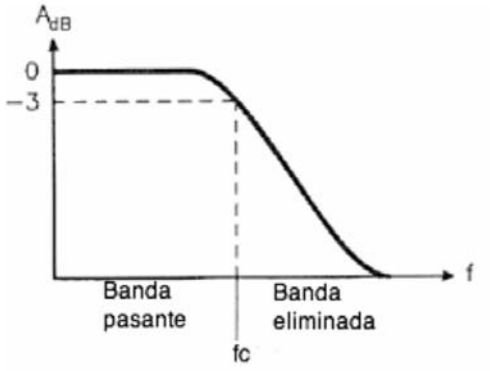

Figura 5.5: Respuesta en frecuencia de un filtro paso bajo.

Los filtros tienen un orden, que indica el grado de rechazo o aceptación a frecuencias por arriba o debajo de la frecuencia de corte, es decir, entre mayor sea el orden del filtro paso bajo mayor será la atenuación de las señales de alta frecuencia conforme la frecuencia aumenta arriba de f_c. La función de transferencia de un filtro paso bajo de segundo orden es la siguiente:

$$H(S) = \frac{H_0\omega_0^2}{S^2 + 2\xi\omega_0 S + \omega_0^2} \tag{5.4}$$

Donde:

H_0 es la ganancia del filtro.

ω_0 es la frecuencia natural.

ξ el factor de amortiguamiento.

El filtro de Butterworth es utilizado para producir la respuesta más plana que sea posible hasta la frecuencia de corte, o bien, para mantener la salida constante casi hasta la frecuencia de corte. El filtro que se va a implementar es un filtro paso bajo de Butterworth de cuarto orden, compuesto por dos etapas de segundo orden. La función de transferencia normalizada de un filtro paso bajas de Butterworth de cuarto orden es la siguiente:

$$H(S) = \frac{1}{S^2 + 0{,}765366S + 1} \cdot \frac{1}{S^2 + 1{,}8477585S + 1} \tag{5.5}$$

La función de transferencia de un filtro paso bajas de cuarto orden con una frecuencia de corte ω_c es:

$$H(S) = \frac{\omega_c^2}{S^2 + 1{,}8477585\omega_c S + \omega_c^2} \cdot \frac{\omega_c^2}{S^2 + 0{,}765366\omega_c S + \omega_c^2} \tag{5.6}$$

5.3.1. El filtro LMF100

Este filtro es un circuito integrado que contiene dos filtros de capacitores conmutados independientes, que con dos a cuatro resistencias externas cada uno puede realizar la función de un filtro de primer o segundo orden, o bien, un filtro de cuarto orden si se conectan los dos filtros en cascada. Para más detalles de este circuito integrado consultar su hoja de datos [6].

5.3.2. Diseño del circuito de filtrado

El filtro de cuarto orden se conseguirá con una conexión en cascada de los dos filtros de segundo orden del LMF100, como se aprecia en la figura 5.6.

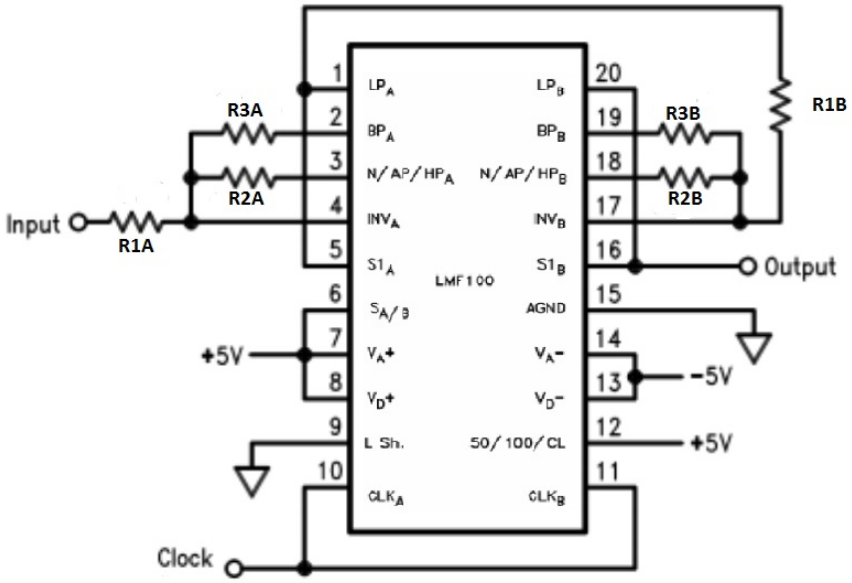

Figura 5.6: Circuito del filtro de cuarto orden utilizando el CI LMF100.

Acorde con la hoja de datos [6], se utiliza el LMF100 en su modo de operación como un filtro paso bajo.

Los parámetros necesarios para configurar el filtro quedarán definidos por las siguientes expresiones:

$$H_{lp} = -\frac{R2}{2R1} \tag{5.7}$$

$$Q = \frac{R3}{R2}\sqrt{2} \tag{5.8}$$

$$f_0 = \frac{f_{clk}}{50}\sqrt{2} \tag{5.9}$$

Donde:

H_{lp} es la ganancia del filtro.

Q el factor de calidad del filtro.

f_0 es la frecuencia de corte.

f_{clk} la frecuencia de la señal de reloj utilizada por el LMF100 para fines de su funcionamiento.

Tales parámetros se pueden obtener de la función de transferencia del filtro (ecuación 5.6), donde:

$$f_0 = \frac{1}{2\pi}\sqrt{\omega_0^2} \tag{5.10}$$

$$Q = \frac{1}{2\xi} \tag{5.11}$$

De acuerdo con la funcón de transferencia del filtro a implementar (ecuación 5.6) y utilizando las expresiones 5.7 y 5.8, los valores respectivos para cada filtro de segundo orden de la figura 5.6 están dados conforme a la siguiente tabla:

	Etapa A	**Etapa B**
Función de transferencia	$H(S) = \frac{\omega_c^2}{S^2+1{,}8477585\omega_c S+\omega_c^2}$	$H(S) = \frac{\omega_c^2}{S^2+0{,}765366\omega_c S+\omega_c^2}$
H_{lp}	1	1
Q	0.54	1.3
R1	51 kΩ	25 kΩ
R2	100 kΩ	51 kΩ
R3	39 kΩ	47 kΩ

Tabla 5.1: Tabla con los valores de los elementos resistivos utilizados para lograr el filtro de cuarto orden deseado.

El utilizar un filtro de Butterworth de cuarto orden brinda la ventaja de que sólo se requiere una señal de reloj para ambos filtros de segundo orden conectados en serie, pues de acuerdo con la función de transferencia de cada filtro de segundo orden la frecuencia de corte es la misma.

5.3.3. Determinación de la frecuencia de corte del filtro paso bajo

La frecuencia de corte del filtro f_c depende de la frecuencia de muestreo de la señal que se desea sea generada, esta última a su vez depende del número de muestras y el periodo utilizado en un momento dado.

La frecuencia de muestreo f_m está dada por la siguiente expresión:

$$f_m = \frac{1}{T_m} \tag{5.12}$$

Recordando que T_m es el periodo de muestreo calculado con la expresión 4.1 Como se desea eliminar el denominado $efecto\ escalera$ presente entre cada muestra, generado en la conversión del DAC, se requiere que se eliminen aquellas discontinuidades presentes con una frecuencia mayor a la f_m. Sin embargo, el filtro empleado no es ideal por lo que la eliminación de los componentes de alta frecuencia se dará en una frecuencia mayor a la f_c del filtro. Por lo anterior, y por recomendaciones tomadas de la hoja de datos del filtro empleado [6], la f_c quedará definida por la siguiente expresión:

$$f_c = \frac{1}{2T_m} \tag{5.13}$$

5.4. Amplificador de salida del filtro

Como el filtro se diseñó con ganancia unitaria, a la salida de éste se tendrá una señal con una amplitud de 4 [V_{pp}]. La señal será ajustada a un valor de 10 [V_{pp}] que es el nivel máximo de la señal a generar por el GSPFP. Esto se logra con otro amplificador operacional ahora configurado como no inversor (figura 5.7), cuya ecuación que describe su ganancia es la siguiente:

$$V_{out} = \left(1 + \frac{R2}{R1}\right) V_{in} \tag{5.14}$$

Si se elige R2=150 kΩ y R1=100 kΩ, la ganancia de esta etapa es de 2.5, por lo que el nivel de la señal a la salida del amplificador será de los 10 [V_{pp}] deseados.

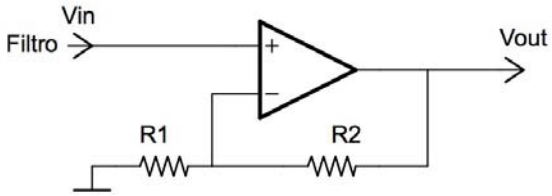

Figura 5.7: Circuito amplificador a la salida del filtro.

5.5. El oscilador programable LTC6904

Anteriormente se mencionó que el filtro de capacitancias conmutadas necesita de una señal de reloj para su funcionamiento, cuya frecuencia es un múltiplo de la frecuencia de corte del filtro paso bajo en un momento dado. Esta señal puede generarse de diferentes maneras; con el propio MCU, con un circuito oscilador formado con transistores o utilizando circuitos integrados muy conocidos como el LM555, sin embargo para cada valor de la frecuencia central de un filtro deseado corresponde un valor diferente de la frecuencia de la señal de reloj.

En el caso de querer generar dicha señal con el MCU, estaríamos limitados a las características del propio MCU en cuanto a la frecuencia máxima que podría generarse. Si se utilizaran circuitos osciladores con transistores o con el propio LM555, se tiene la limitante de que para diferentes frecuencias requeridas necesitaríamos cambiar algún valor de una resistencia o de un capacitor que conforman al circuito, lo que resulta impráctico si se desea tener un dispositivo compacto. He aquí la importancia de tener un oscilador programable como el LTC6904, que se explicará a continuación.

El LTC6904 es un oscilador digital de alta precisión ajustable mediante el bus IIC, cuyo funcionamiento se explicó en el capítulo 2. Una característica importante de este dispositivo es que no necesita de componentes externos para su funcionamiento. A continuación se mencionan las características más importantes del LTC6904, tomadas de su hoja de datos [4]

- Oscilación variable entre 1 kHz y 68 MHz.
- Programable mediante IIC.
- Operación en un rango de 2.7 [V] a 5.5 [V].

En la figura 5.8 se observa un diagrama de conexiones del LTC6904, controlado por el MCU presente en la tarjeta MINICON_08GT a través de las líneas SDA y SCK del bus IIC. Se distingue un cambiador de nivel que se explicará más adelante. A la salida CLK se obtiene la señal de reloj de 5 [V] con la frecuencia para la cual fue programado el circuito, que sirve como entrada al filtro LMF100.

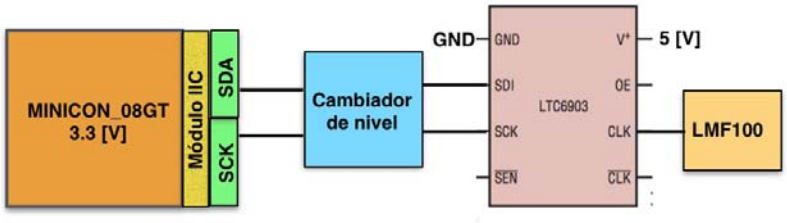

Figura 5.8: Diagrama de conexiones del oscilador programable con la tarjeta MINICON_08GT.

La frecuencia de la señal que se desea generar es determinada por la siguiente ecuación:

$$f = 2^{OCT} \cdot \frac{2078 \quad [Hz]}{2 - \frac{DAC}{1024}} \tag{5.15}$$

Donde:

f es la frecuencia de la señal de reloj.

OCT representa un valor digital de 4 bits (0 -15 en notación decimal).

DAC representa un valor digital de 10 bits (0 - 1023 en notación decimal).

Primeramente se tiene que calcular el valor de OCT, utilizando la siguiente ecuación y redondeando el valor al entero más cercano:

$$OCT = 3{,}322 \log\left(\frac{f}{1039}\right) \tag{5.16}$$

Posteriormente se calcula el valor de DAC mediante la siguiente ecuación:

$$DAC = 2048 - \frac{2078[Hz] \cdot 2^{(10+OCT)}}{f} \tag{5.17}$$

Es importante recordar que estos valores son determinados en el software de la PC y enviados por puerto serie al MCU, el cual a su vez los envía al

oscilador utilizando el protocolo IIC.

En la figura 5.9 se muestra la forma en que el MCU debe transmitir los datos al LTC6904 siguiendo el protocolo IIC. En total son transmitidos 3 bytes, o bien 24 bits, que se conforman de la siguiente manera: dirección del dispositivo a controlar (bits 0:5), bit de dirección (bit 6), bit de escritura (bit 7), valor de OCT (bits 8:11), valor de DAC (bits 12:21) y bits de configuración de la salida (bits 22:23).

Los bits de configuración de la salida se configuran de tal forma que la salida CLK del LTC6904 esté activada.

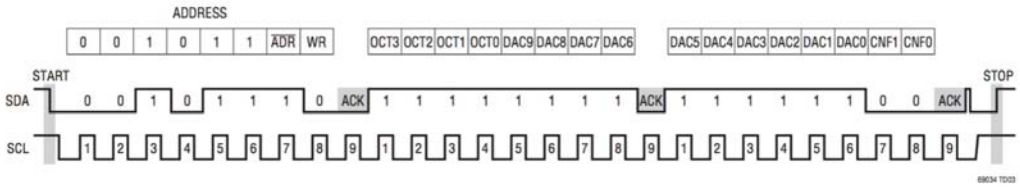

Figura 5.9: Envío de bytes de configuración al OPIIC.

Hasta este punto se explicó el funcionamiento básico y la configuración del LTC6904 para fines del proyecto del GSPFP. Si se desea profundizar sobre el funcionamiento de este dispositivo, se recomienda consultar la hoja de datos [4].

5.5.1. Cambiador de nivel bidireccional para el bus IIC

De acuerdo con las características eléctricas del MCU éste trabaja con 3.3 [V], mientras que el LTC6904 tiene un rango de trabajo de 2.7 [V] a 5.5 [V]. Por otro lado, la señal de reloj que requiere el LMF100 para un correcto funcionamiento debe tener un nivel lógico de 5 [V], entonces el LTC6904 debe ser polarizado con 5 [V].

Al trabajar con niveles lógicos de 3.3 [V] y 5 [V] para el MCU y el oscilador LTC6904 respectivamente, es necesario un cambiador de nivel bidireccional en las líneas del bus IIC, para que la comunicación por el bus pueda establecerse entre los dos dispositivos con diferentes niveles lógicos.

El adaptador de nivel bidireccional mostrado en la figura 5.10 consiste en un transistor MOSFET para la línea SDA y SCL del bus IIC.

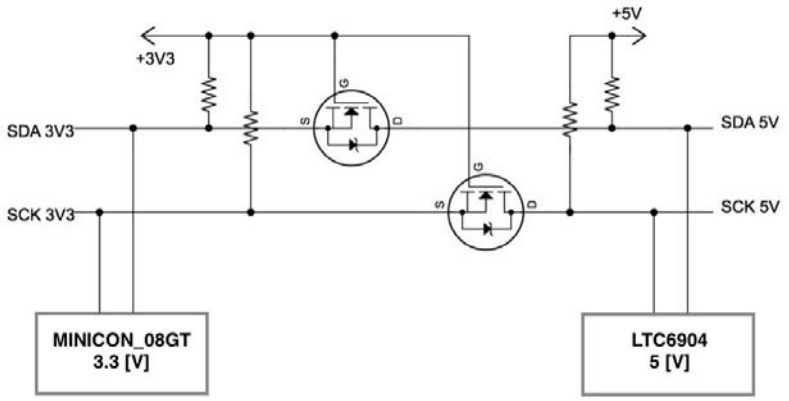

Figura 5.10: Circuito cambiador de nivel bidireccional para el bus IIC.

Como se puede deducir en la figura 5.10, en el sector izquierdo se alimenta con 3.3 [V] por lo que corresponde al MCU, mientras que el sector derecho se alimenta con 5 [V] que corresponde al LTC6904. Cada dispositivo cuenta con sus resistencias de pull-up requeridas, conectados a la fuente de alimentación que le corresponde a cada dispositivo.

Funcionamiento del cambiador de nivel

Para explicar su funcionamiento, se mencionan los 3 estados posibles del circuito:

- El primer estado es cuando no hay actividad en el bus, es decir, ningún dispositivo pone en cero las líneas SDA o SCL. En este caso, el diodo de protección integrado en cada transistor impedirá que exista conducción de la zona con 5 [V] a la zona con 3.3 [V], lo cual brinda protección al dispositivo de baja tensión. La tensión en la puerta del MOSFET (G) será la misma que la de la fuente (S), por lo que no habrá diferencia de potencial entre G-S y consecuentemente no habrá conducción en ninguno de los transistores. Además, ambas líneas del bus mantendrán estado alto gracias a las resistencias de pull-up.

- El segundo estado posible es cuando hay un estado bajo en una línea del lado de 3.3 [V]. La fuente (S) del transistor pasará a un estado bajo mientras que la puerta (G) permanecerá con una tensión de 3.3 [V],

generando una diferencia de potencial que iniciará la conducción del transistor. En esta situación, la línea del lado de alta tensión también será inducida a un estado bajo y será arrastrada por la conducción del MOSFET.

- El tercer estado se da cuando a la línea que corresponde a 5 [V] el dispositivo la pone en estado bajo. Inicialmente el diodo entrará en conducción desde la línea de 3.3 [V] hacia la tensión de 0 [V] de la otra sección puesta en estado bajo. Por la conducción del diodo la fuente (S) pasa a un estado bajo y, en consecuencia, se vuelve a producir una diferencia de potencial entre la puerta (G) y la fuente (S) del transistor, comenzando la conducción a través del MOSFET y asegurando que ambos lados del bus estén en estado bajo.

5.6. Circuito para ajustar la amplitud de la señal

La señal a la entrada del circuito (v_{in}) de la figura 5.11 se encuentra en +5 [V] y -5 [V], que será el valor de amplitud máximo de la señal bipolar generada. Para ajustar a un valor menor de amplitud de la señal se utiliza nuevamente un amplificador operacional en configuración inversora.

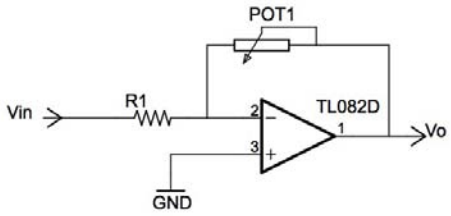

Figura 5.11: Circuito para ajustar la amplitud de la señal.

La expresión que determina la ganancia es la misma que la ecuación 5.3, con la diferencia que utiliza el potenciómetro POT1, por lo que la expresión quedará definida como:

$$V_0 = -\frac{R_{POT1}}{R1} V_{in} \tag{5.18}$$

Si se elige el valor de R1 = 100 kΩ y POT1=100 kΩ, al variar el valor del potenciómetro se puede ajustar la amplitud de la señal en un rango de

$0[V_p]$ a $5[V_p]$, o bien de $0[V_{pp}]$ a 10 $[V_{pp}]$, de acuerdo con el valor que el usuario proponga en la interfaz de la PC.

5.7. Circuito para ajustar el nivel de offset

El nivel de offset es el voltaje de corriente directa sobre el cual está montada la señal generada. Para añadir un nivel de offset a la señal, se utilizó el circuito de la figura 5.12.

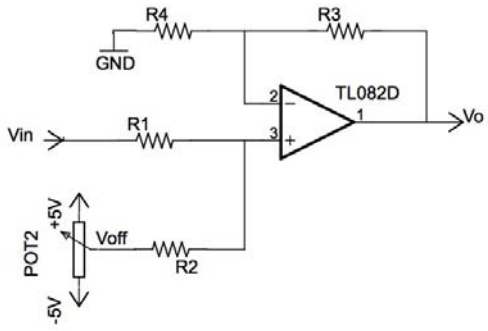

Figura 5.12: Circuito sumador para agregar la tensión de offset..

El circuito de la anterior figura es un circuito sumador, el cual le sumará a la señal de entrada V_{in} la tensión de corriente directa que proporciona el POT2, que está conectado a +5[V] y -5[V] en sus terminales. La expresión que determina el valor de V_0 a la salida del amplificador es la siguiente:

$$V_0 = \frac{R3}{R1}V_{in} + \frac{R3}{R2}V_{off} \tag{5.19}$$

Si se elige el valor de las resistencias R1=R2=R3=R4, la anterior expresión queda de la siguiente forma:

$$V_0 = V_{in} + V_{off} \tag{5.20}$$

De acuerdo con la anterior expresión, la señal a la salida será igual a la señal de la entrada más V_{off}, que es la tensión en corriente directa que otorga POT2, comprendida entre +5[V] y -5[V]. De esta forma es como se le añade un nivel de offset deseado a la señal generada, definido por el usuario en el software que corre en la PC.

5.8. El potenciómetro digital DS1267

Para lograr que la amplitud y el nivel de offset de la señal puedan ser controlados desde la interfaz de usuario del software de la PC, es necesario que los potenciómetros POT1 y POT2 de los circuitos de las figuras 5.11 y 5.12 respectivamente, puedan manipularse de forma digital a través de dicho software mientras la señal se está desplegando. La solución es utilizar un potenciómetro digital programable.

El DS1267 es un chip que contiene dos potenciómetros digitales independientes controlados por el bus SPI, explicado en el capítulo 3. El diagrama de bloques del DS1267 tomado de su hoja de datos [5] se puede observar en la figura 5.13.

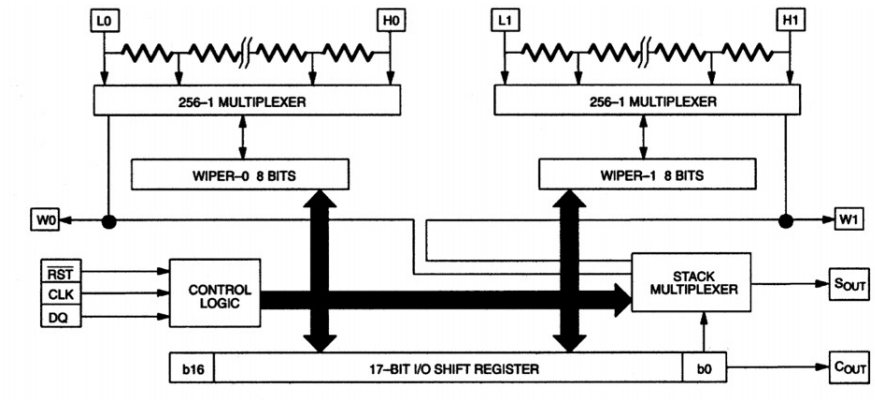

Figura 5.13: Diagrama de bloques del POTSPI.

Cada potenciómetro se compone de 256 secciones resistivas, entre cada sección resistiva existe un punto que es accesible a la flecha del potenciómetro. La posición de la flecha del potenciómetro es ajustada por un valor de 8 bits, es decir, por un valor comprendido entre 0 y 255, recibido a través del bus SPI.

Los valores de 8 bits de cada potenciómetro son escritos a un registro de corrimiento de 17 bits, que es usado para almacenar las dos posiciones de la flecha de cada potenciómetro y un bit que permite que se interconecten en cascada los dos potenciómetros.

5.8.1. Determinación del valor de POT1

El potenciómetro de la figura 5.11 se usa para cambiar la amplitud de la señal a la entrada del amplificador, cuya ganancia está regida por la ecuación 5.18. El valor de la resistencia que se puede ajustar con el POT 1 es de 0 a 100 kΩ, entonces, si el registro que lo controla es de 8 bits se tiene la siguiente relación:

$$\text{Registro} = 0 \rightarrow R_{POT1} = 0\ \Omega$$
$$\text{Registro} = 255 \rightarrow R_{POT1} = 100\ \text{k}\Omega$$

De la figura 5.11 se tiene que R1=100 kΩ, y de acuerdo a la ecuación 5.18, la señal de 5 $[V_p]$ reducirá su amplitud conforme R_{POT1} disminuya. Con lo anterior se puede determinar una relación para obtener el valor que se le asigna al registro del POT1, de acuerdo con la amplitud de la señal que el usuario especifica.

$$V_{P1} = \frac{255}{5\,[V]} V_{amp} \tag{5.21}$$

Donde:

V_{P1} es el valor en notación decimal de los 8 bits que configuran al POT1.

V_{amp} es el valor de la amplitud de la señal comprendido entre 0 [V] y 5 [V], especificado por el usuario.

5.8.2. Determinación del valor de POT2

El potenciómetro POT2 presente en el circuito de la figura 5.12 es utilizado para obtener una señal de corriente directa, que se le sumará a la señal deseada para modificar el nivel de offset definido por el usuario. El potenciómetro también podía verse verse como en la figura 5.14.

Si se observa la anterior figura se nota la presencia de un factor K, que es un factor para determinar el valor de la resistencia de POT2 de acuerdo con V_{off}, que es el voltaje de offset deseado comprendido entre +5[V] y -5[V]. Haciendo un análisis aislado de este pequeño circuito, se puede determinar el valor del registro de 8 bits del POT2 necesario para obtener el V_{off} definido por el usuario en el software ejecutable en la PC.

Resolviendo el circuito de la figura 5.14, por superposición, el voltaje de offset quedará definido como la siguiente expresión:

$$V_{off} = V_f\,(1 - 2K) \tag{5.22}$$

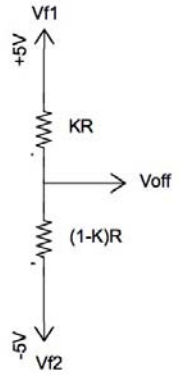

Figura 5.14: Circuito cambiador de nivel bidireccional para el bus IIC.

Donde Vf = Vf1 = -Vf2 = 5[V]. Despejando el factor K:

$$K = \frac{1}{2}\left(1 - \frac{V_{off}}{V_f}\right) \tag{5.23}$$

Por lo tanto, la expresión final que determina el valor de V_{P2} que es el valor en notación decimal del registro de 8 bits que controla al POT2, está dada por:

$$V_{P2} = 255 \cdot K \tag{5.24}$$

5.8.3. Inversor para la línea SS del bus SPI

En el capítulo 3 se explicó el protocolo estándar de transmisión de datos por SPI, en el que se menciona que para iniciar la comunicación la línea SS debe llevarse a un nivel lógico igual a cero. Esa transición la hace automáticamente el MCU, sin embargo, por características del diseño del DS1267 se tiene que hacer lo contrario, es decir, la línea SS debe llevarse de un nivel bajo a un nivel alto para iniciar la comunicación. Para solucionar ese inconveniente se diseñó el circuito de la figura 5.15, que hace la función de una compuerta lógica NOT.

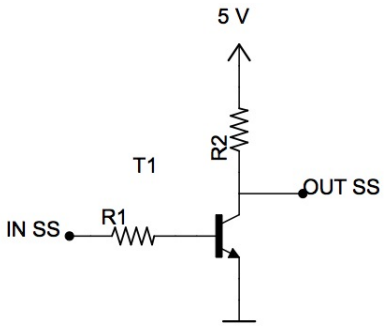

Figura 5.15: Circuito inversor para la línea SS del bus SPI.

El circuito cuenta con dos resistencias y un transistor NPN. El principio de su funcionamiento es el siguiente: Cuando La línea IN SS está en estado alto, habrá conducción en el transistor y en consecuencia, la salida OUT SS estará en estado bajo. Cuando la línea IN SS está en estado bajo el transistor no conduce, por lo que la salida OUT SS estará en estado alto.

6 | Pruebas de la funcionalidad básica del prototipo

En este capítulo se mostrarán algunas pruebas realizadas con el GSPFP construido. En el capítulo 1 se mostró el diagrama con los bloques funcionales del prototipo, por lo que se mostrarán los resultados obtenidos en el diseño y prueba de cada etapa.

En la figura 6.1 se muestra un ejemplo del aspecto que tendría en la interfaz de usuario una forma de onda determinada que se desea sea generada por el GSPFP en un momento dado. En la interfaz, como ya se ha mencionado se especificaría además, el periodo de la señal (T), el número de muestras por periodo (nmp), la amplitud y el offset para la especificación de la misma. La forma de onda mostrada está simplemente ahí para fines ilustrativos de la funcionalidad del software manejador que corre en la computadora PC.

Los parámetros que son requeridos al usuario se aprecian en la figura 6.1, y éstos son T, nmp, amplitud y offset. Acorde con la ejemplificación mostrada los valores de los parámetros son respectivamente 250 [μs], 50, 1.5 [V_p] y 2 [V]. Como se explicó en capítulos anteriores, el periodo (del que depende la frecuencia) depende del valor de nmp. Para esta ejemplificación se utilizó un nmp bajo, con la finalidad de obtener una frecuencia alta (dentro del rango de frecuencias del GSPFP) y de observar con mayor detalle el funcionamiento de la etapa de filtrado.

Una vez que el usuario ha especificado gráficamente una determinada forma de onda así como también los parámetros mencionados en el párrafo anterior, para que esta señal sea generada por el dispositivo se requerirá al usuario que oprima el botón *bajar señal* . Una vez hecho esto el software manejador generará una tabla homologada asociada con la señal a generar, de modo que ésta presente valores extremos de +10 y -10 [V] respectivamente. La tabla es un arreglo de bytes porque el DAC es de 8 bits. Después de esto el software manejador enviará la tabla vía puerto serie byte por byte

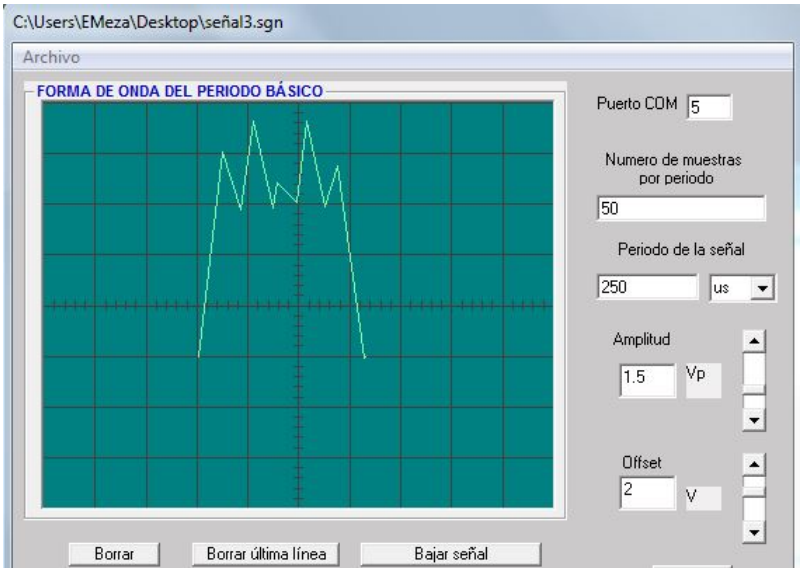

Figura 6.1: Aspecto de una forma de onda en la IU.

hacia el MCU de la tarjeta MINICON_08GT que es la CPU de la interfaz de reproducción. Una vez que ha terminado el envío, el propio software de la interfaz de reproducción reproduce este en el DAC cíclicamente con la temporalidad adecuada a las especificaciones definidas por el usuario. Para fines de los parámetros analógicos asociados con la señal a generar (amplitud y offset) la señal primitiva es procesada por la etapa de adecuación analógica que está integrada por circuitos módulos funcionales analógicos entre otros, amplificadores operacionales y potenciómetros ajustables digitalmente.

Cabe señalar que desde el punto de vista del usuario, para la generación de una determinada señal, bastará especificar ésta como se muestra en la figura 6.1, para después de esto oprimir el botón *bajar señal*, después de lo cual deberá aparecer en la salida del dispositivo la señal con las características definidas por el propio usuario en el software manejador.

6.1. Formas de onda presentes en la etapa de adecuación analógica (EAA)

Para fines ilustrativos se muestran aquí las diversas formas de onda presentes en la etapa de adecuación analógica al generarse la señal especificada.

La figura 6.2 muestra la señal de 20 [V_{pp}] a la salida del DAC para la ejemplificación mostrada en la figura 6.1.

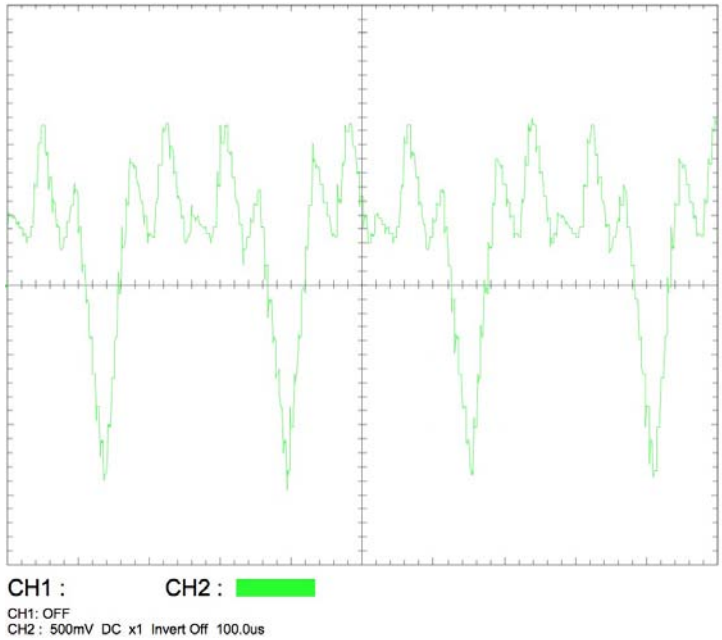

Figura 6.2: Vista en el osciloscopio de la señal de prueba a la salida del DAC.

En la figura 6.3 se muestra la salida del bloque de la EAA identificado como atenuador. Nótese que los valores extremos de la señal de salida del atenuador son +2 [V] y -2 [V], lo cual se requiere para la funcionalidad correcta del filtro paso bajas. Además de atenuar la señal al nivel deseado, se observa que también la invierte debido a la configuración en la que se utilizó el circuito del amplificador operacional mostrado en la figura 5.4 del capítulo 5.

La señal de la figura 6.3 es la entrada a la siguiente etapa, que es un filtro paso bajas de capacitancias conmutadas, cuyas especificaciones de respuesta en frecuencia son calculados por el software manejador generando éste lo propio para que la señal de reloj del filtro sea la requerida por la especificación funcional de éste. La etapa de filtrado consta de un filtro paso bajo de cuarto orden, logrado con dos filtros de segundo orden con ganancia unitaria y con una configuración inversora para cada filtro, por lo que a la salida la señal filtrada tendrá una amplitud de 4 [V_{pp}] y también saldrá invertida, como se aprecia en la figura 6.4.

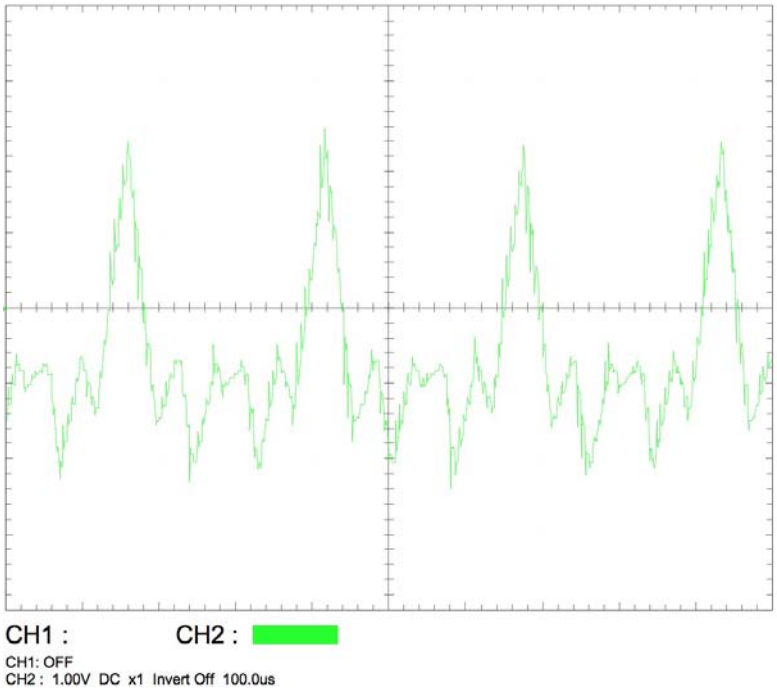

Figura 6.3: Vista en el osciloscopio de la señal de prueba a la salida del atenuador.

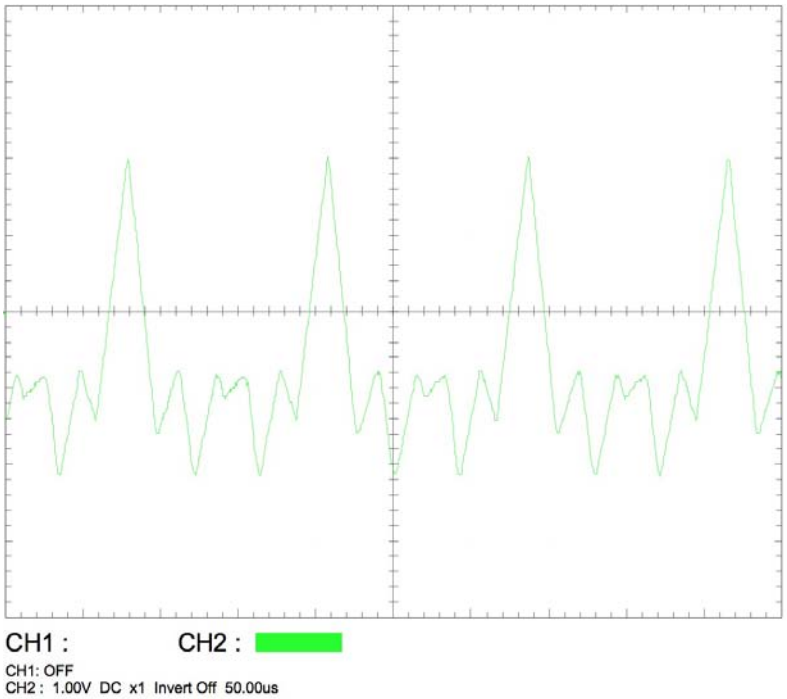

Figura 6.4: Vista en el osciloscopio de la señal de prueba a la salida del filtro paso bajo.

La siguiente etapa de la EAA denominada amplificador ajusta la señal de la salida del filtro con una ganancia igual a 2.5, esta vez a una amplitud de 10 $[V_{pp}]$ que es el valor máximo pico a pico para el GSPFP. Si se observa el circuito de la figura 5.7 del capítulo 5, se utiliza una configuración no inversora del amplificador operacional, por lo que a la salida de esta etapa la señal será de $10[V_{pp}]$ y seguirá invertida como se observa en la figura 6.5.

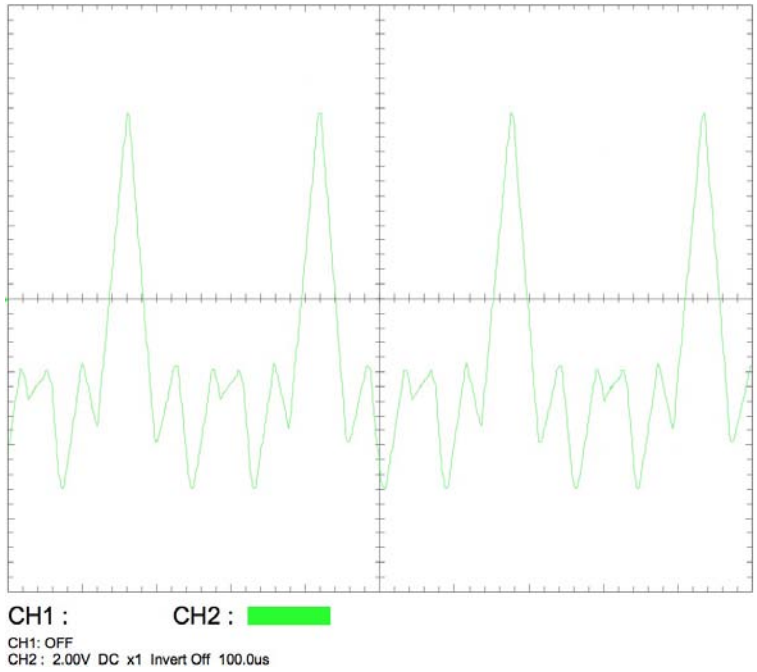

Figura 6.5: Vista en el osciloscopio de la señal de prueba a la salida del bloque amplificador.

Con la señal obtenida en el bloque anterior, sólo queda llevar la señal a los niveles de amplitud y offset requeridos, que para el ejemplo eje de este capítulo son 1.5 $[V_p]$ y 2 $[V]$ respectivamente. En el circuito de la figura 5.11 corresponde al bloque ajustador de amplitud y se observa que para atenuar la señal según se requiera, se utiliza nuevamente un amplificador operacional en configuración inversora, por lo que a la salida se tendrá la señal con la amplitud de 1.5 $[V_p]$ y ya no seguirá invertida. La señal obtenida tras pasar por el circuito que se encarga de ajustar la señal al nivel de amplitud que se especificó en la IU se puede apreciar en la figura 6.6.

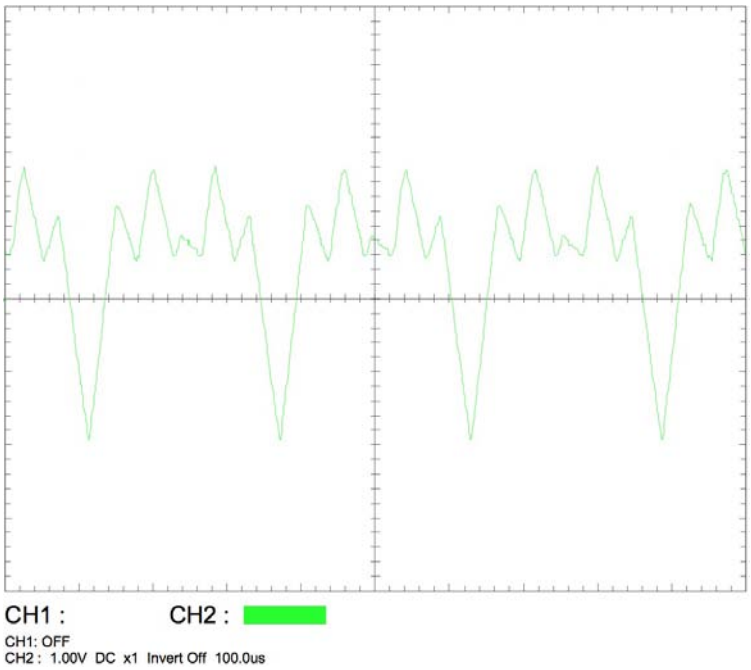

Figura 6.6: Vista en el osciloscopio de la señal de prueba con la amplitud requerida.

Por último, la señal pasa por el circuito de la figura 5.12 del capítulo 5 correspondiente al bloque ajuste de offset, que es un amplificador operacional configurado como sumador no inversor. A la señal del bloque anterior que le corresponde a la figura 6.6 se le suma un voltaje de corriente directa de 2 $[V]$, obteniendo finalmente la señal con las características definidas previamente por el usuario.

La señal que se obtuvo tras sumarle el offset correspondiente a este ejemplo se aprecia en la figura 6.7, que es la señal de salida del GSPFP. Si se analiza a detalle, la señal tiene una amplitud de 3 $[V_{pp}]$, o bien 1.5 $[V_p]$, un offset de 2 $[V]$ y un periodo de 250 $[\mu s]$ que coreesponde a una frecuencia de 4000 Hz, tal y como se especificó en la IU.

Si se observan las figuras 6.2 y 6.7 que son respectivamente la señal a la salida del DAC y la de la salida propia del GSPFP se puede apreciar que desaparecen las continuidades finitas o tambien denominado $efecto\ escalera$ del DAC, gracias a la etapa de filtrado. Cabe destacar que se puede obtener la misma forma de onda con una mejor definición aumentando el nmp, lo cual significaría reducir la frecuencia.

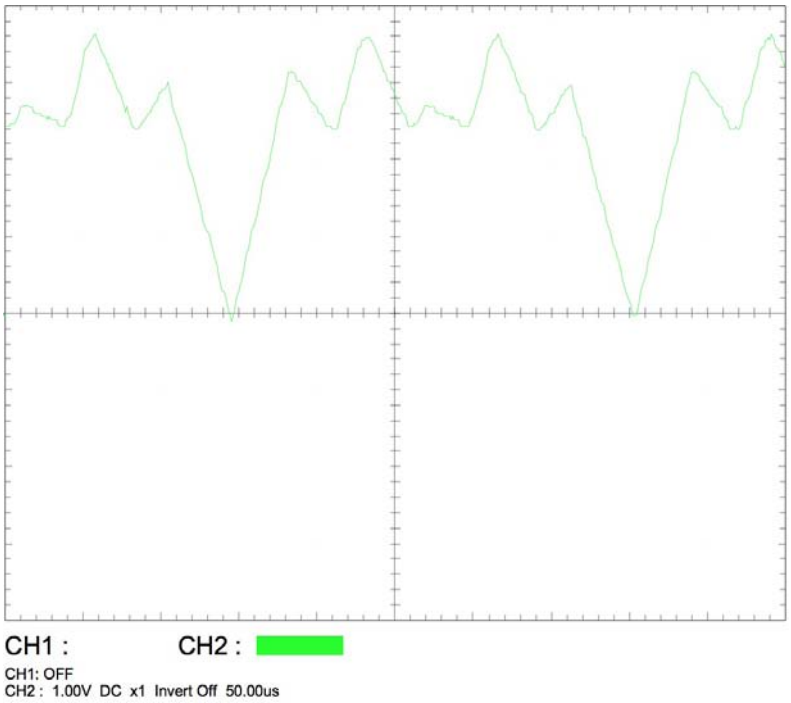

Figura 6.7: Vista en el osciloscopio de la señal de prueba con el offset requerido.

7 | Ejemplos de aplicación

Se ha hablado anteriormente que el GSPFP puede generar casi cualquier forma de onda y que la periodicidad de ésta está limitada por el número de muestras por periodo que se desee tenga la señal en un momento dado. Con estas condiciones, será posible realizar la forma de onda de señales electrocardiográficas, dado que la frecuencia de estas es relativamente baja.

El poder generar este tipo de señales se aprecia que el GSPFP tiene aplicaciones potenciales de como instrumento auxiliar en ingeniería biomédica. Por ejemplo, el dispositivo puede emplearse para la generación de señales electrocardiográficas para fines de simulación o bien didácticos en cuanto a la caracterización y análisis de este tipo de formas de onda. Para fines ilustrativos, a continuación se simulará con el GSPFP la señal de un electrocardiograma normal y de algunas anomalías cardiacas con la frecuencia real, en la que todas sus partes son identificables.

Es importante mencionar que las señales electrocardiográficas estrictamente no son consideradas señales periódicas, pues si se analiza la señal en cierto intervalo de tiempo es posible que ésta sufra alteraciones o un comportamiento abrupto, sin embargo para periodos cortos de tiempo pueden tratarse como tal.

La información referente a los electrocardiogramas que fueron tomados para los siguientes ejemplos se obtuvo de [7].

7.1. Generación de la forma de onda de un electrocardiograma normal

En la figura 7.1 se observa el trazado de un electrocardiograma normal, en la que sus componentes son claramente distinguibles. Un latido cardiaco normal consiste en una onda P, un complejo QRS y una onda T.

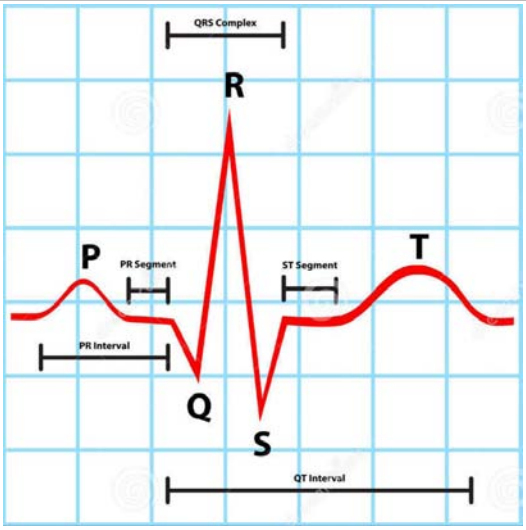

Figura 7.1: Trazado de un electrocardiograma normal.

Cada componente de la señal tiene un periodo, y sus valores normales quedan comprendidos conforme a la siguiente tabla:

Parte de la onda	**Duración (s)**
Onda P	Menos de 0.11
Intervalo PR	Entre 0.12 y 0.20
Onda Q	Menos de 0.04
Intervalo QR	Menos de 0.05
Onda S	Menos de 0.04
Intervalo QRS	Menos de 0.1
Intervalo QT	Entre 0.35 y 0.45

Tabla 7.1: Duración de las partes de la onda de un electrocardiograma normal

La imagen 7.2 muestra la pantalla virtual de la IU en la que se dibujó la forma de la señal, se especificó el número de muestras igual a 500 y el periodo de 820 [ms]. Los parámetros de amplitud y offset se establecieron en 1 [V_p] y 0 [V] respectivamente.

En la imagen 7.3 se muestra la señal producida por el GSPFP vista en un osciloscopio digital. Se puede apreciar que todas sus partes son claramente identificables, y si se mira más a detalle, cada parte de la onda cumple con los valores presentados en la tabla 7.1. Midiendo el periodo entre picos de la

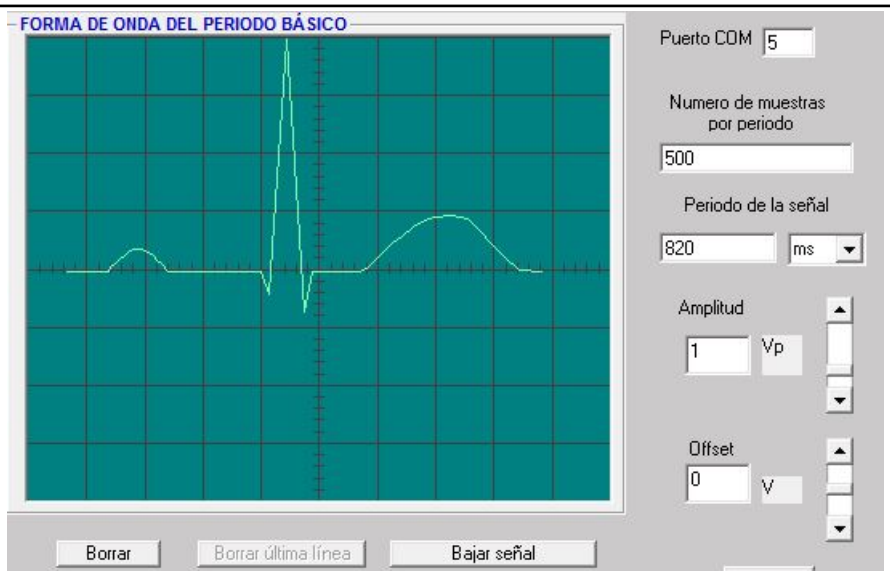

Figura 7.2: Electroardiograma normal en la pantalla virtual de la IU.

onda R, éste corresponde a los 820 [ms] definidos en la IU, lo que equivale a un frecuencia cardiaca de 73 latidos por minuto, que se encuentra dentro del rango normal para un electrocardiograma.

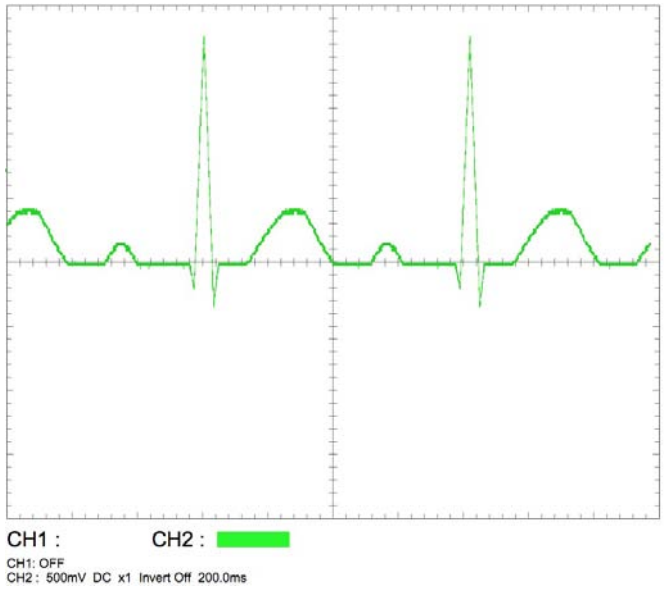

Figura 7.3: Vista en el osciloscopio de la señal de un electrocardiograma normal producida por el GSPFP.

7.2. Generación de ondas de anomalías del ritmo cardiaco

A continuación se mostrarán las señales que el GSPFP puede generar para representar algunas anomalías que se pueden presentar en el ritmo cardiaco, con la finalidad de mostrar el amplio rango de formas de onda que que se pueden simular.

La figura 7.4 muestra la señal eléctrica de una taquicardia ventricular, la frecuencia normal de este ritmo oscila entre 140 y 200 latidos por minuto y en la que se observa que el complejo QRS es ancho y de forma extraña. Para el ejemplo se consideró un periodo de 375 [ms] que equivale a una frecuencia cardiaca de 160 latidos por minuto.

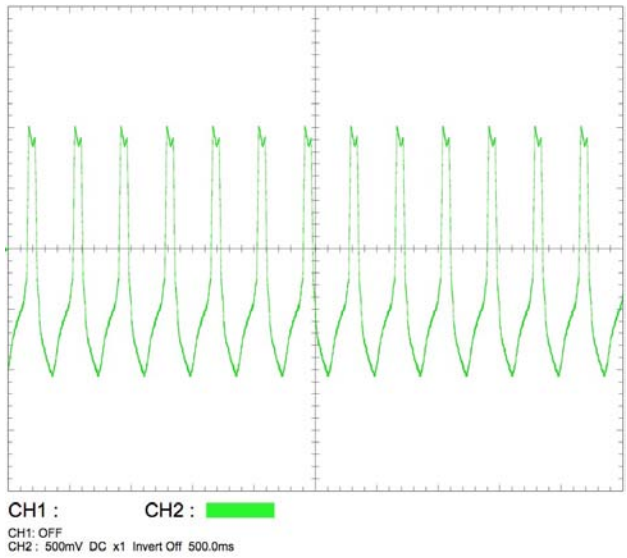

Figura 7.4: Vista en el osciloscopio de la señal eléctrica que simula una taquicardia ventricular.

La figura 7.5 muestra la señal de un ritmo idioventricular. Se puede observar que el complejo QRS es similar al de una taquicardia ventricular por lo que es común que se confundan entre sí, sin embargo se distinguen por la forma de su onda P y por su frecuencia, ya que la frecuencia de este ritmo está comprendida entre 30 y 40 latidos por minuto. En el ejemplo se especificó un periodo de 1.71 [s] que equivale aproximadamente a una frecuencia

cardiaca de 35 latidos por minuto.

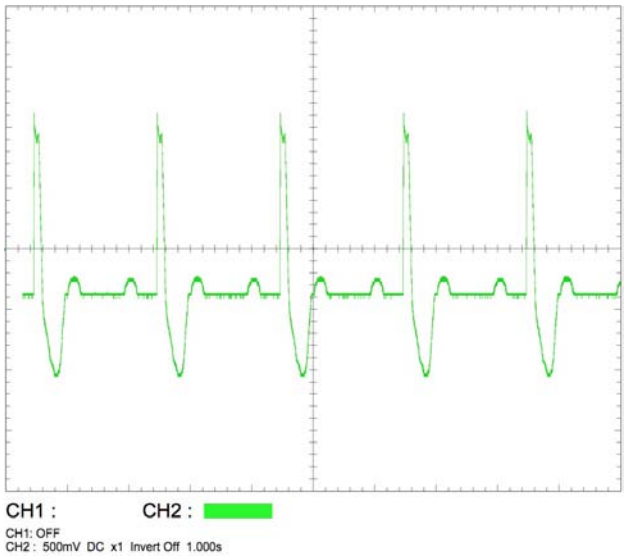

Figura 7.5: Vista en el osciloscopio de la señal eléctrica que simula un ritmo idioventricular.

Con los ejemplos anteriores se comprueba la importancia que podría llegar a tener el GSPFP para la enseñanza, en este caso al generar las señales se podría enseñar a los alumnos a identificar entre las diferentes anomalías, ya sea por alguna diferencia en la forma de onda, o en casos en los que existen anomalías con formas de onda muy similares, con la frecuencia de la señal.

8 | Conclusiones

En este trabajo se diseñó el prototipo funcional de un generador de señales periódicas de forma predeterminada, que entre sus posibles aplicaciones, la principal es servir como un instrumento didáctico para los laboratorios de electrónica y señales, pues de esta manera los alumnos cuentan con un instrumento tangible que pueden manipular para entrar en contacto con lo que se ve en la teoría.

Las aplicaciones del GSPFP no sólo tienen que ver con áreas de la ingeniería, sino también en áreas como la medicina, pues como se vio en el capítulo 8, se pueden generar señales de un electrocardiograma con la frecuencia real. Por lo anterior mencionado, el GSPFP podría convertirse en un instrumento didáctico para el área médica.

Durante el desarrollo del trabajo se presentaron diversos inconvenientes para lograr objetivos específicos, sin embargo gracias a que el MCU cuenta con diversos módulos internos para la comunicación con dispositivos externos, estos inconvenientes pudieron ser resueltos con resultados satisfactorios. Tal es el caso del filtrado de la señal, ya que hacerlo con filtros activos convencionales era inviable por la cantidad de componentes necesarios para filtrar señales de diferentes frecuencias. Gracias a un filtro de capacitancias conmutadas y a que el MCU cuenta con un módulo IIC, se logró reemplazar la función de los filtros activos, incluso con mayor precisión.

Las características de los elementos utilizados para el diseño del prototipo limitaron algunas de las características propias del GSPFP. La frecuencia de la señal y velocidad de transmisión fueron limitadas por características del MCU, la amplitud y offset fueron limitados por las características eléctricas de los componentes electrónicos. No obstante, se puede acondicionar la señal con circuitos externos dependiendo de lo que requiera la aplicación.

La interfaz de usuario creada permite definir las características de la señal

y además brinda las facilidades de un programa convencional, sin embargo es posible que se puedan realizar mejoras y actualizaciones. Por ejemplo, se podría modificar la interfaz de usuario para poder ser ejecutado en múltiples plataformas, o bien, se puede actualizar el programa para que sea capaz de leer los datos de las señales que entrega un electrocardiógrafo para posteriormente reproducirlas.

Actualmente existen en el mercado equipos comerciales con la desventaja de que resultan muy costosos. Se cumplió el objetivo de crear un dispositivo de bajo costo, y sobre todo, se cumplió el objetivo de diseñar este dispositivo como una aplicación directa de material creado en la Facultad de Ingeniería de la UNAM.

De manera general, se lograron los objetivos planteados al inicio de este trabajo y se demostró que el ambiente de desarrollo AIDA08, que incluye el software PUMMA08_EST y la tarjeta MINICON_08GT, es un sistema robusto y completo para el desarrollo de proyectos de control e instrumentación.

A | Código del programa ejecutable en el MCU de la MINICON_08GT

```
'Programa ejecutable en el MCU de la MINICON_08GT
'GSPFP

dim byacad(1024) as integer
'direcciones de memoria en BASIC
defvarbptr ptad &h00
defvarbptr ptadd &h03
defvarbptr tpm1sc &h30
defvarbptr tpm1modh &h33
defvarbptr tpm1modl &h34
defvarbptr spic1 &h28
defvarbptr spic2 &h29
defvarbptr spibr &h2a
defvarbptr spis &h2b
defvarbptr spid &h2d
defvarbptr iic1a &h58
defvarbptr iic1f &h59
defvarbptr iic1c &h5a
defvarbptr iic1s &h5b
defvarbptr iic1d &h5c

'direcciones de memoria en ensamblador
        iniens
        tpm1sc@ equ $30
        ptad@ equ $00
        tpm1modh@ equ $33
        spis@ equ $2b
        spid@ equ $2d
        iic1c@ equ $5a
        iic1s@ equ $5b
        iic1d@ equ $5c
        iic1f@ equ $59

'parámetros iic
        mov #$56,iic1f@
        mov #$88,iic1c@
        finens

         ptadd=&hff      'pta salidas
         spic1=&h52      'habilita spi, mcu maestro
         spic2=&h19      'spi como salida
         spibr=&h77      'baud rate
         iic1a=&h2e      'dirección del disp. a controlar
         iic1f=&h56      'velocidad de transmisión

'********** Captura nmp% **********
         nmp%=0
         gosub cap_byte
         iniens
         lda capbte~
         sta nmp%
         finens

         gosub cap_byte
         iniens
         lda capbte~
         sta nmp%+1
       '  sta #$00
         mov #$02,$00
         finens
```

```
'***************** Captura xr& ******************
            xr&=0
            gosub cap_byte
            iniens
            lda capbte~
            sta xr&+2
            finens

            gosub cap_byte
            iniens
            lda capbte~
            sta xr&+3
            mov #$03,$00
            finens
'*****************************************************
           for i%=0 to nmp%-1
'********** Captura de byte presente a colocar en entrada de DAC **************

            bycad%=0
            gosub cap_byte

            iniens
            lda capbte~
            sta bycad%+1
            finens
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
           byacad(i%)= bycad%       '12.75*(bycadr+10.)
           next i%
'*************************************************************************
           iniens
finiic:     brset 5,iic1s@,finiic      'verifica si está ocupado el bus iic
           bset 4,iic1s@
           bset 5,iic1c@    'MCU maestro
           bset 4,iic1c@    'MCU transmite

'transmite dirección del opiic
           lda #$2e
           sta iic1d@
           jsr txiic

'lee y envía primer byte
           jsr lee#car
           sta iic1d@
           jsr txiic

'lee y envía segundo byte
           jsr lee#car
           sta iic1d@
           jsr txiic

           bclr 5,iic1c@     'señal de paro
           bra fin

'subrutina transmisión por iic
txiic:      brclr 1,iic1s@,txiic
            bset 1,iic1s@
            brset 0,iic1s@,fintx
fintx:      rts
fin:        nop
           finens
```

```
'... ……Reciclaje de tabla en DAC .....
'actualiza variables

          nmpd%=nmp%
          nmpdtes%=nmpd%
          apuntab%=0

           iniens

           ldhx #byacad     'h:x<-- dirini de tabla
           sthx apuntab%  '(apuntab%) <-- dirini de tabla
           ldhx nmpdtes%
           sthx nmpd%      '(nmpd%)<-- n˙mero de muestras por periodo

          mov #$04,$00

'lee y configura el valor del Pe a utilizar
          jsr lee#car
          cmp #$01
          beq preuno
          cmp #$40
          beq preses

preuno:     mov #$48,tpm1sc@
             bra continua
preses:     mov #$4e,tpm1sc@

continua:   ldhx xr&+2
          sthx tpm1modh@
          cli
          finens

'++++++++++Reset del MCU o cambio de amplitud y offset+++++++++++++++
otron:
          iniens
          jsr lee#car         'lee comando para determinar acción
          cmp #$01
          beq uno              'salto para restablecer el MCU
          cmp #$00
          beq cero            'salto a configuración de potSPI
          bra noo

uno:      bclr 6,tpm1sc@              'limpia interrupción
          mov #$00,ptad@               'limpia puerto
          jmp $fe7d                   'receptor de comandos
          bra noo

cero:       lda #$00          'primer byte potSPI
txspi1:     brclr 5,spis@,txspi1
            sta spid@
            jsr lee#car       'lee segundo byte
txspi2:     brclr 5,spis@,txspi2
             sta spid@
             jsr lee#car      'lee tercer byte
txspi3:     brclr 5,spis@,txspi3
            sta spid@

          bra noo

noo:        nop
```

```
        finens
        goto otron

' Subrutina cap_byte
' Espera la recepción de un byte por el puerto serie
' Al retornar
' capbte~ <-- byte recibido

cap_byte:    capbte~=0

        iniens
        jsr lee#car
        sta capbte~
        finens

        return

'+++++Rutina de interrupción por overflow del MCU+++++++++
'Los números son el número de ciclos de reloj de cada instrucción

servovf:

        iniens
        psha
        lda tpm1sc@                 '3
        bclr 7,tpm1sc@              '5

        ldhx apuntab%               '5
        lda $01,x  'bycad%+1        '3
        sta ptad@                   '3
        aix #$02                    '2
        sthx apuntab%               '5
        ldhx nmpd%                  '5
        aix #$ff                    '2
        sthx nmpd%                  '5
        cphx #$0000                 '3
        bne ciclot                  '3

        ldhx #byacad 'h:x<-- dirini de tabla
        sthx apuntab% '(apuntab%) <-- dirini de tabla
        ldhx nmpdtes%
        sthx nmpd% '(nmpd%)<-- n`mero de muestras por periodo
ciclot:   pula                      '1    45
        finens
        retint

        dataw &hdfee servovf
```

B | Código del software manejador ejecutable en la PC

```
Form1 - 1

'Declaración de variables
Dim col As Long
Dim m(100), b(100), mtra(1000) As Single
Dim datoabre, nombre As String
Dim cambio, FGon As Boolean
Dim xp(100), yp(100), mult As Single
Dim byalnmp, bybajnmp, byalxr, manda, bybajxr, nmp, oct, dac, byaliic, bybajiic As Integer
Dim xant, byacad(1000), txoff, txamp, nps, Pe As Integer
Dim ampli, offse, k As Single
Dim per, Tm As Double
Dim nc, iic, frec, fclk As Long
'Función para transmitir por puerto serie
Private Sub txsi(a)
MSComm1.Output = Chr$(a)
End Sub
'cálculo y transmisión de los valores del opiic
Private Sub iicosc()
'frecuencia de corte del filtro
frec = Int(1 / (2 * Tm))
'frecuencia de la señal de reloj
fclk = Int(frec * 50 / Sqr(2))
MsgBox "frec " & fclk
oct = Int(3.322 * Log(fclk / 1039) / Log(10))
dac = Int(2048 - ((2078 * 2 ^ (10 + oct)) / fclk))
'concatenación de los 2 bytes
iic = oct * 4096 + dac * 4
byaliic = Int(iic / 256)
bybajiic = iic Mod 256
MsgBox "byal " & byaliic
MsgBox "bybaj " & bybajiic
txsi (byaliic)
txsi (bybajiic)

End Sub
'Cálculo de los valores de los potspi y transmisión
Private Sub ampoff()
txtamp.Text = ampli
txtoff.Text = offse
k = (1 - offse / 5) / 2
txoff = CInt(255 - k * 255)
txamp = CInt(255 - ((ampli * 255) / 5))
txsi (0)
txsi (txamp)
txsi (txoff)
'MsgBox "amp " & txamp
End Sub
'detectar cambios en la señal
Private Sub cambioOK()
If cambio = True Then
    resp = MsgBox("¿Desea guardar el archivo abierto?", 3)
    If resp = 6 Then
        If nombre = "" Then
            mnuguardarc_Click
        ElseIf nombre <> "" Then
            mnuguardar_Click
        End If
    ElseIf resp = 2 Then
        FGon = True
        Exit Sub

    ElseIf resp = 7 Then

    End If
End If
End Sub
'protocolo pumma, salto a dirección del memoria del programa en MCU
Private Sub enviar()
txsi (14)
txsi (1)
txsi (0)
txsi (1)
txsi (2)
'Código de jmp $182c en modo extendido
txsi (204)
txsi (24)
txsi (44)
'cálculo de bytes componentes de nmp%
```

```
Form1 - 2

byalnmp = Int(nmp / 256)
bybajnmp = nmp Mod 256
'envío de bytes componentes de nmp%
txsi (byalnmp)
txsi (bybajnmp)
MsgBox "nc=" & nc
'-----------------------------------
'cálculo de bytes componentes de xr
byalxr = Int(nc / 256)
bybajxr = nc Mod 256
'envío de bytes componentes de xr
txsi (byalxr)
txsi (bybajxr)
'-------------------------------------
MsgBox "voy a transmitir tabla"
'envío de los datos der la tabla homologada
For k = 0 To nmp - 1
  'MsgBox "bycad= " & byacad(k)
  txsi (Int(byacad(k)))
Next k
'llamado a funciones para enviar datos de opiic Pe y potspi
iicosc
txsi (Pe)
ampoff
MsgBox "tabla transmitida"
End Sub
'al presionar botón bajar señal
Private Sub cmdbaja_Click()
'multiplicador dependiendo de unidad seleccionada
If Combo1.ListIndex = 0 Then
mult = 0.000001
ElseIf Combo1.ListIndex = 1 Then
mult = 0.001
ElseIf Combo1.ListIndex = 2 Then
mult = 1
End If
'cálculo del Tm
Tm = (per * mult) / (nmp - 1)
'MsgBox "Tm " & Tm
If Tm < 0.000005 Then
MsgBox "El valor mínimo para el periodo entre muestras es 5 microsegundos, modifique los valores de
 nmp y periodo", 48, "Señal inviable"
Exit Sub
End If
'seleccionar Pe del timer
If Tm >= 0.000005 And Tm <= 0.001 Then
Pe = 1
ElseIf Tm > 0.001 Then
Pe = 64
End If
'cálculo número de cuentas NC
nc = Tm / (0.00000005 * Pe)
'Cálculo de los coeficientes de la ecuacion de recta
For i = 0 To nps - 2
  m(i) = (yp(i + 1) - yp(i)) / (xp(i + 1) - xp(i))
  b(i) = yp(i) - (m(i) * xp(i))
Next i
'-------------------------------------------------------
'espacio entre muestras
Em! = (xp(nps - 1) - xp(0)) / (nmp)
'-------------------------------------------------------
av% = 0
cuenta! = 0
'Cálculo de cada muestra en el eje X
For i = 0 To nmp - 1

       If xp(0) + cuenta! >= xp(av% + 1) Then
        av% = av% + 1
        mtra(i) = m(av%) * (xp(0) + cuenta!) + b(av%)
        cuenta! = cuenta! + Em!

      ElseIf xp(0) + cuenta! < xp(av% + 1) Then
        mtra(i) = m(av%) * (xp(0) + cuenta!) + b(av%)
        cuenta! = cuenta! + Em!
       End If

'valores límites
```

```
Form1 - 3

If i = 0 Then
    valmax! = mtra(i)
    valmin! = mtra(i)
End If

    If mtra(i) >= valmax! Then
        valmax! = mtra(i)

    ElseIf mtra(i) <= valmin! Then
        valmin! = mtra(i)
    End If

Next i
'cálculo del valor del byte de cada muestra
For i = 0 To nmp - 1
        byacad(i) = Int((255 / (valmax! - valmin!)) * (mtra(i) - valmin!))
Next i

res = MsgBox("Tabla creada correctamente. ¿Desea bajar la señal?", 1, "Bajar a Gen")
If res = 1 Then
    enviar
ElseIf res = 2 Then
    Exit Sub
End If
End Sub
'borrar señal de la pantalla virtual
Private Sub cmdborra_Click()
Picture1.Cls
xant = 0
nps = 0
Dibuja
cambio = False
End Sub
'borrar último tramo de recta dibujado
Private Sub cmdborralin_Click()
col = &H808000

If nps = 2 Then
    Picture1.Line (xp(nps - 2), yp(nps - 2))-(xp(nps - 1), yp(nps - 1)), col
    xant = 0
    nps = nps - 2
    cmdborralin.Enabled = False
    Exit Sub
End If
cambio = True
Picture1.Line (xp(nps - 2), yp(nps - 2))-(xp(nps - 1), yp(nps - 1)), col
Picture1.PSet (xp(nps - 1), yp(nps - 1)), col
'actualiza datos
xant = xp(nps - 2)
nps = nps - 1
End Sub
'comando para restablecer el MCU
Private Sub cmdreset_Click()
txsi (1)
End Sub
'forma
Private Sub Form_Load()
Combo1.ListIndex = 0
Form1.BackColor = RGB(203, 202, 202)
Label1.BackColor = RGB(203, 202, 202)
Label3.BackColor = RGB(203, 202, 202)
Label2.BackColor = RGB(203, 202, 202)
lbloff.BackColor = RGB(203, 202, 202)
lblamp.BackColor = RGB(203, 202, 202)
cmdborralin.Enabled = False

'Inicialización elemental del puerto com
' Use COM1.
    MSComm1.CommPort = 5
    ' 9600 baud, no parity, 8 data, and 1 stop bit.
    MSComm1.Settings = "9600,n,8,1"
    ' Tell the control to read entire buffer when Input is used.
    MSComm1.InputLen = 0
    ' Open the port.
    MSComm1.PortOpen = True

Form1.Caption = "Nuevo.sgn"
```

```
Form1 - 4

'CommDial.FileName = "Nuevo"
nombre = ""
Dibuja
'default
ampli = 5
offse = 0
End Sub
'dibujado de la pantalla virtual
Private Sub Dibuja()
Picture1.MousePointer = 2
Dim i As Integer
'Dibuja lineas verticales
For i = 1 To 10
 xv = i * 550
 Picture1.Line (xv, 0)-(xv, 4400)
Next i

'Dibuja lineas horizontales
For i = 1 To 8
 yh = i * 550
 Picture1.Line (0, yh)-(5500, yh)
Next i

'lineas centrales vertical
For i = 1 To 40
 yv1 = i * 110
 Picture1.Line (2700, yv1)-(2800, yv1)
Next i

'lineas centrales horizontal
For i = 1 To 50
 xv1 = i * 110
 Picture1.Line (xv1, 2250)-(xv1, 2150)
Next i
'escala y origen de la pantalla virtual
Picture1.Scale (0, 4400)-(5500, 0)

End Sub
'abrir archivo .sgn
Private Sub mnuabrir_Click()
cambioOK
If FGon = True Then
FGon = False
Exit Sub
End If
CommDial.CancelError = True
On Error GoTo IsOK

CommDial.Filter = "Archivo señal (*.sgn)|*.sgn"
CommDial.ShowOpen
IsOK:
If Err = 0 Then

    abretabla = FreeFile

    Open CommDial.FileName For Input As #abretabla
    Line Input #abretabla, datoabre
    cmdborra_Click
    nps = CInt(datoabre)

    For i = 0 To nps - 1
        Line Input #abretabla, datoabre
        xp(i) = CDbl(datoabre)
        Line Input #abretabla, datoabre
        yp(i) = CDbl(datoabre)
        If i > 0 Then
        Picture1.Line (xp(i - 1), yp(i - 1))-(xp(i), yp(i)), RGB(133, 250, 175)
        End If
        If EOF(abretabla) Then
        Close #abretabla
        End If
    Next i
    xant = xp(nps - 1)
    Close #abretabla

Form1.Caption = CommDial.FileName
nombre = CommDial.FileName
```

```
cambio = False

ElseIf Err = cdlCancel Then
Else
    MsgBox "Error Inesperado"
End If
End Sub
'guardar archivo .sgn en uno existente
Private Sub mnuguardar_Click()
   If nombre = "" Then
   mnuguardarc_Click

   Else
   tabla = FreeFile
    Open nombre For Output As #tabla
    Print #tabla, nps
    For i = 0 To nps - 1
        Print #tabla, xp(i)
        Print #tabla, yp(i)
    Next i
    Close #tabla
    End If
    cambio = False
End Sub
'guardar nuevo archivo
Private Sub mnuguardarc_Click()
CommDial.CancelError = True
On Error GoTo IsOK

 CommDial.Filter = "Archivo Señal (*.sgn)|*.sgn"
 CommDial.ShowSave

IsOK:
If Err = 0 Then
    tabla = FreeFile
    Open CommDial.FileName For Output As #tabla
    Print #tabla, nps
    For i = 0 To nps - 1
        Print #tabla, xp(i)
        Print #tabla, yp(i)
    Next i
    Close #tabla
    Form1.Caption = CommDial.FileName
    nombre = CommDial.FileName
    cambio = False
ElseIf Err = cdlCancel Then

Else
    MsgBox "Error Inesperado"
End If

End Sub
'nuevo archivo
Private Sub mnunuevo_Click()
cambioOK
If FGon = True Then
FGon = False
Exit Sub
End If

cmdborralin.Enabled = False
cmdborra_Click
nombre = ""
Form1.Caption = "Nuevo.sgn"
CommDial.FileName = nombre
End Sub
'salir del programa
Private Sub mnusalir_Click()
cambioOK
If FGon = True Then
FGon = False
Exit Sub
End If
MSComm1.PortOpen = False
End
End Sub
'colocación de puntos en la pantalla
```

Form1 - 6

```
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
cambio = True
xact = X
If xact <= xant Then
    MsgBox "Punto incorrecto"
    Exit Sub
End If
Picture1.PSet (X, Y), RGB(133, 250, 175)
xant = X
xp(nps) = X
yp(nps) = Y
'trazado de rectas
If nps > 0 Then
    cmdborralin.Enabled = True
    Picture1.Line (xp(nps - 1), yp(nps - 1))-(xp(nps), yp(nps)), RGB(133, 250, 175)
End If
nps = nps + 1
End Sub


Private Sub Text3_Change()
car = Text3.Text
If IsNumeric(car) Then
car1 = Int(car)
Else
MsgBox "introduzca un valor correcto"
Text3.SelStart = 0
Text3.SelLength = Len(Me.Text3.Text)
End If
End Sub
'captura de nmp
Private Sub txtnmp_Change()
If IsNumeric(txtnmp.Text) Then
nmp = CInt(txtnmp.Text)
    If nmp > 512 Then
    MsgBox "El número máximo numero de muestras es 512"
    txtnmp.Text = ""
    nmp = 0
    End If
Else
txtnmp.Text = ""
End If

End Sub
'captura del periodo
Private Sub txtper_Change()
If IsNumeric(txtper.Text) Then
per = CDbl(txtper.Text)
Else
txtper.Text = ""
End If
End Sub
'cambio de amplitud
Private Sub vscamp_Change()
ampli = vscamp.Value / 10
ampoff
End Sub
'cambio de offset
Private Sub vscoff_Change()
offse = vscoff.Value / 10
ampoff
End Sub
```

Bibliografía

[1] FREESCALE, *MC9S08GB60A Data Sheet* , Rev. 2, 2008. Archivo MC9S08GB60A.pdf descargable desde la página www.freescale.com

[2] Salvá, A., *AIDA08, Ambiente Integrado para Desarrollo y Aprendizaje con Microcontroladores de la Familia 68HC908 de Freescale (Guía de Usuario)*, 2011. Archivo muaida08ve2011.pdf descargable de la página http: //dctrl.fi-b.unam.mx/∼salva

[3] Salvá, A. y Altamirano, L., *Dispositivos Chipbas8, microcontroladores HC08 programables en lenguaje BASIC*, SAAEI, Madrid España, 2009.

[4] LINEAR, *LTC6904, 1 kHz to 68 MHz Serial Port Programmable Oscillator*, 2003. Archivo 69034fe.pdf descargable de la página www.cds.linear.com

[5] MAXIM INTEGRATED, *DS1267, Dual Digital Potentiometer Chip Data Sheet*. Archivo DS1267.pdf descargable desde la página www.maximintegrated.com

[6] TEXAS INSTRUMENTS, *LMF100 Dual High-Performance Switched Capacitor Filters*, July 1999, Rev. June 2015, Archivo lmf100.pdf descargable desde la página www.ti.com

[7] Wartak, Joseph, *Interpretación de Electrocardiogramas*, Interamericana, México D.F., 1977.

[8] Summerville, D., *Embedded Systems Interfacing for Engineers Using the Freescale HCS08 Microcontroller Part I: Assembly Language Programming*, Morgan & Claypool, USA, 2009.

[9] Summerville, D., *Embedded Systems Interfacing for Engineers Using the Freescale HCS08 Microcontroller Part II: Digital and Analog Hardware Interfacing*, Morgan & Claypool, USA, 2009.

[10] Pereira, F., *HCS08 Unleashed, Designer's Guide to the HCS08 Microcontrollers*, BookSurge Publishing, USA, 2008.

[11] Graeme, J., Tobey, G. and Huelsman, L., *Operational Amplifiers Design and Applications*, McGRAW-HILL KOGAKUSHA, Tokyo Japan, 1971.

[12] Coughlin, R. y Driscoll, F., *Circuitos Integrados Lineales y Amplificadores Operacionales*, Segunda edición, Prentice-Hall, México, 1987.

[13] Kopka, H., Daly P.W., *A Guide to LaTeX*, Addison-Wesley, Reading, MA, 1999.