**Universidad Tecnológica Nacional**

**Facultad Regional Villa María**

****

**Cátedra:** Medidas Electrónicas II

**Profesores:** Ingeniero Franco Salvático.

Ingeniero José Luis Catalano.

**Integrantes:** Griffa Joaquín, Martino Ignacio, Vico Matías

**Especialidad:** Ing. Electrónica

**Fecha:** 17/02/13

**Tema:**

**Generador de señales con conectividad USB**

**Consigna:**

**Cátedra:** Medidas Electrónicas II y Técnicas Digitales III.

**Tema:** Calculo, diseño, implementación, y medición de generador sinusoidal digital con comunicación USB.

**Objetivo:**

* Aprender como diseñar un generador digital sinusoidal.
* Aprender el funcionamiento de la comunicación USB.

**Materiales e instrumentos necesarios**:

1. Osciloscopio
2. Analizador lógico
3. Frecuencímetro

**Especificaciones del modulador**

1. Frecuencia 100Hz a 10kHz
2. Amplitud 1-4V
3. Resolución 10 bits
4. DAC Integrado
5. Conector de salida tipo SIP para bus de datos y sinusoide

**Especificaciones del puerto USB:**

1. Tipo de comunicación del USB: Bulk.

**Desarrollo y Mediciones a presentar:**

1. Diagrama esquemático circuital, circuito impreso y código en caso de usar microcontrolador.
2. Medición de frecuencia mínima y máxima.
3. Medición de amplitud mínima y máxima.
4. Medición distorsión armónica en 1kHz.
5. Medición de bus de datos con analizador lógico para corroborar secuencia binaria.

**Introducción:**

El proyecto fue desarrollado para cumplir con la condición de regularización de las materias de Medidas Electrónicas II y Técnicas Digitales III.

En la primera, la consigna es el desarrollo de un generador de señales senoidales. Mientras que en la restante es la comunicación mediante conectividad USB.

Por lo tanto para cumplir con ambas consignas, lo que se hizo fue desarrollar un generador de señales con conectividad USB.

Para ello, fueron necesarios la utilización de 2 microcontroladores PIC. Esto fue necesario, debido a que uno de los requisitos era la generación de onda senoidal con una frecuencia máxima de 10 KHz y no era posible realizarlo con solo uno de ellos, porque no se podía alcanzar dicha frecuencia. Entonces se utilizó el (18F4550), el cual es el encargado de realizar la generación de la señal y el otro (18F14K50) es el responsable de la comunicación USB y de enviarle la información correcta al otro PIC para le generación de dicha señal.

PC

PIC

18F14K50

PIC

18F4550

DAC

CARGA

ALIMENTACIÓN

8

EUSART

USB

GENERACIÓN

**Contenido Teórico:**

Básicamente se lo puede dividir en dos partes bien diferenciadas: la generación de señales y la conectividad USB.

* **Conectividad USB:**

**Partes necesarias para un desarrollo basado en un microcontrolador PIC:**

1. Programa de la aplicación para el microcontrolador (PIC).
2. Programa de la aplicación para la PC.
3. Edición de las librerías suministradas por Microchip.
4. Consideraciones para la programación del microcontrolador PIC:

Los microcontroladores PIC de la serie 18 (que es la utilizada para esta aplicación) que soportan la comunicación por USB se denominan genéricamente 18F**XX**5**X** (en nuestro caso 18F4550).

Incluir los siguientes archivos (suponiendo que se utiliza el compilador proporcionado por PIC C):

* pic\_usb.h → Serie 16.
* pic18\_usb.h → Serie 18.
* usb.h → Definiciones comunes y prototipos.
* usb.c → Manejo del stack USB, la interrupción USB y el pedido de configuración del endpoint de configuración (0).
* usb\_cdc.h → Incluye los archivos anteriores para generar un dispositivo USB CDC.

Interrupciones válidas:

* int\_usb() → Un evento USB ha ocurrido (requiere atención del programa). CCS maneja automáticamente el control de la interrupción.

Funciones relevantes de las librerías:

* usb\_init() → Inicializa el hardware USB. Debe esperar infinitamente para que el dispositivo USB se conecte al BUS, esto no significa que el USB sea enumerado por el HOST. Hace uso de la interrupción USB.
* usb\_init\_cs() → Realiza la misma función que usb\_init() pero no espera a que el dispositivo se conecte al BUS. Esto es útil para un dispositivo que no es alimentado por el BUS y puede operar sin una conexión USB.
* usb\_task() → Si se utiliza el sensado de conexión y usb\_init\_cs() para la inicialización esta función, debe ser continuamente revisada para verificar por una posible conexión al BUS. Cuando se desconecte del BUS resetéa el stack USB y el periférico. Hace uso de la interrupción USB. DEBE definirse USB\_CON\_SENSE\_PIN en la aplicación para referír este pin
* usb\_detach() → Remueve al PIC del BUS. Esta función es llamada automáticamente por usb\_task() si se pierde la conexión con el BUS pero puede ser llamada manualmente.
* usb\_attach() → Conecta el PIC al BUS. Es llamada automáticamente por usb\_task() pero puede ser llamada automáticamente.
* usb\_attached() → Si se usa el pin de sensado, se devuelve un TRUE si este se encuentra en alto, de lo contrario se devuelve FALSE.
* usb\_enumerated() → Devuelve TRUE si el dispositivo fue enumerado por el HOST. Si esta etapa se realizó, la conexión se encuentra en modo normal de operación y se permite el envío y recepción de paquetes.
* usb\_put\_packet(endpoint, data, len, tgl) → Coloca un paquete de datos en el buffer del endpoint establecido. Devuelve TRUE si hay éxito, retorna FALSE si el buffer aún está lleno con el último paquete.
* usb\_puts(endpoint, data, len, timeout) → Envía información al endpoint seleccionado. Se diferencia de usb\_put\_packet() en que va a enviar múltiples paquetes si la información no entra en un solo paquete.
* usb\_kbhit(endpoint) → Devuelve TRUE si el endpoint especificado tiene información en su BUS de recepción.
* usb\_get\_packet(endpoint, ptr, max) → Lee bytes hasta una cantidad especificada por “max” del endpoint especificado y lo almacena en el puntero especificado por “ptr”. Devuelve el número de bytes almacenados en ptr.
* usb\_gets(endpoint, prt, max, timeout) → Lee el mensaje disponible en el endpoint especificado. La diferencia con usb\_get\_packet() es que esta instrucción en cambio espera hasta la recepción completa del mensaje donde este puede contener más de un paquete. Devuelve el número de bytes recibidos.

1. Consideraciones para la programación de la PC:

Tener en cuenta que el programa debe estar vinculado a la dll de Microchip en el caso de usar su driver por medio de la **mpusbapi.dll**.

Para cada plataforma de PC (esto es basado en su arquitectura y el SO implementado en esta) es necesario contar con el driver correspondiente, no es lo mismo tener un driver para 2 arquitecturas (x32 y x64) o SO diferentes. Es recomendable tener el del SO más nuevo y de la arquitectura más compleja (Windows 7 y 64 bits respectivamente). Se puede obtener de la página de Microchip bajo el nombre de “USB Framework”.

1. Descriptores:

Son estructuras de datos que le permiten al HOST aprender sobre el dispositivo. Cada descriptor contiene información del dispositivo como un todo o sobre una parte específica de este. TODOS los dispositivos USB DEBEN responder para los pedidos del HOST sobre los descriptores USB estándar.

Esta información varía y depende de cada dispositivo y de la aplicación particular en la que se utilice. Un descriptor para una configuración típica está compuesto como mínimo por:

* Al menos un descriptor de configuración.
* Uno o más descriptores de interfaz.
* Uno o más descriptores de endpoints.

Puede existir además un descriptor de cadena de texto (USB string descriptor) que contiene una descripción del dispositivo. De esta forma el dispositivo puede pasar esta información descriptiva al SO. Tiene una estructura particular y trabaja en conjunto a otro descriptor de Offset (USB string descriptor offset).

Es en este archivo que también son definidos el Vendor ID (VID) y el Product ID (PID), números que sirven para la identificación de un dispositivo en particular y así también su asociación con un driver determinado para asegurar su funcionamiento.

Microchip entrega dos valores de uso masivo para VID y PID, estos se pueden usar de manera libre y son:

**VID = 04D8h**

**PID = 0011h** o **000Bh**

Existen varios tipos de descriptores que pueden ser de uso obligatorio (o no) y que definen distintos aspectos de funcionamiento del USB del dispositivo. Los mas comunes son los de:

* Dispositivo (obligatorio).
* Configuración (obligatorio).
* String (opcional, texto descriptivo).
* Interfaz (obligatorio).
* Endpoint (obligatorio para usar otro endpoint además del de configuración).

Excepto para dispositivos compuestos, cada dispositivo contiene SOLO un descriptor de dispositivo que contiene la información de este y especifica el número de configuraciones que este soporta. Para cada configuración este deberá contar con un descriptor de configuración con información sobre el consumo de energía del dispositivo, el número de interfaces que esta configuración soporta y para cada interfáz a su vez se deberá contar con un descriptor de interfáz que especifique el número de endpoints en cada una de ellas. Finalmente cada endpoint tiene un descriptor de endpoint que contiene la información necesaria para comunicarse con este. Una interfáz sin un descriptor de endpoint utiliza el endpoint de configuración (0), el dispositivo debe devolverlo asó como también todos los de interfáz, endpoin y otros descriptores subordinados tantos como bytes haya solicitado el HOST. El HOST no puede solicitar solo un endpoint (su descriptor) en particular.

En general la estructura de los descriptores es la misma en cuanto a los datos que contiene y el orden en el que estos se presentan, este es:

* bLength: Byte que da la longitud del descriptor en bytes.
* bDescriptorType: Byte que identifica el tipo de descriptor.

El resto de los campos varían en función del tipo de descriptor (device, configuration, interface, endpoint, string, etc).

**Consideraciones a nivel Hardware del PIC:**

Configuración y consideración de hardware para el clock y oscilador:

* Para el uso del USB con el protocolo 2.0 es necesario que el periférico cuente con una señal de Ck (USB) de 48 MHz. Esto puede lograrse mediante la configuración correcta de ciertos bytes (fuses) que van relacionados al valor de resonador/cristal colocado externamente. Este cristal (según su frecuencia de trabajo) será necesario colocar dos capacitores que estabilizarán la frecuencia del cristal.
* Para el uso del modo de alta velocidad (USB 2.0) es imperioso que el Ck del USB sea configurado a 48 MHz, esto No es necesario para el modo de trabajo en baja velocidad, por otra parte no es necesario que el Ck del CPU sea configurado al mismo que el del USB por lo que el mismo (según su modo de configuración) tendrá una frecuencia mientras que el Ck de la CPU puede tener un valor menor.

Consideraciones de hardware de conexión, estabilización y alimentación del PIC:

* Si se utiliza el periférico USB es obligatorio que la tensión de la patita Vusb sea estabilizada (estabiliza la tensión utilizada por el módulo USB) y lleva como mínimo un capacitor de 220 nF a GND.
* El pin de la alimentación del BUS USB es el encargado de la alimentación del USB. Sin embargo, dependerá del tipo de alimentación del USB. Si el diseño tiene alimentación propia no es necesario conectar Vbus. Por otra parte si se desea que el BUS USB alimente la placa será necesario conectar Vbus a Vdd y de esta forma no se precisará una fuente de alimentación externa. Puede utilizarse un jumper que conecte o abra la conexión entre Vbus y Vdd.

No se recomienda utilizar ambos medios de alimentación.

* Se puede (o no) utilizar el regulador interno de tensión y las resistencias internas de pull-up (del USB) pero deberá solo utilizarse o las internas o una disposición similar externa, pero no ambas.

Registros y flags vinculadas a la conexión USB:

* Registro UCON:el bit USBEN para de 0 a 1 cuando se conecta el PIC al BUS USB (attach). Al desconectar es necesario setear esa bandera a un bajo mediante una integración de actividad del USB. Esto está relacionado a la conexión y desconexión del USB en caliente sin reiniciar el PIC.

**Especificaciones USB del PIC 18F4550**

Características Generales

Este dispositivo cuenta con un motor de interfaz serial (SIE) USB compatible con full-speed y low-speed, que permite una comunicación rápida entre cualquier host USB y el microcontrolador PIC. El SIE puede interconectarse directamente al USB, utilizando el transceptor interno, o puede ser conectado a través de un transceptor externo. Un regulador de 3.3V interno se encuentra disponible para alimentar el transceptor interno, en aplicaciones de 5V.

Se han incluido algunas características especiales de hardware para mejorar el rendimiento. La memoria de puerto dual en el espacio de la memoria de datos del dispositivo (RAM USB) ha sido suministrada para compartir el acceso directo a la memoria entre el núcleo de microcontrolador y el SIE. Se proporcionan también algunos descriptores de búfer, permitiendo a los usuarios programar libremente el endpoint (Buffer de almacenamiento de datos) de uso de la memoria en el espacio de memoria RAM USB.

Se ha proporcionado un Streaming Parallel Port (SPP) para el apoyo de la transferencia ininterrumpida de grandes volúmenes de datos, tales como datos isocronos, a búferes de memorias externas.

Estado y control del USB

El funcionamiento del módulo USB se configura y gestiona a través de tres registros de control. Además, una total de 22 registros se utilizan para administrar las transacciones actuales del USB. Los registros son los siguientes:

* Registro de control USB (UCON)
* Registro de configuración USB (UCFG)
* Registro de estado de transferencia USB (USTAT)
* Registro de dirección de dispositivo USB (UADDR)
* Registros de número de fotograma (UFRMH: UFRML)
* Registros de habilitación de los endpoints, desde el 0 al 15 (UEPN)

Regulador Interno

El PIC 18F4550 posee un regulador interno de 3.3V para proporcionar alimentación al transceptor interno y para los pull-ups internos/externos. Es requerido un condensador externo de 220 nF (± 20%) para la estabilidad de la tensión anteriormente mencionada.

El regulador puede activarse o desactivarse mediante software. Cuando está activado, el voltaje es visible en el pin VUSB. Cuando el regulador está desactivado, una fuente de 3,3V se debe proporcionar a través del pin VUSB para alimentar el transceptor interno.

RAM USB

Los datos USB se mueven entre el núcleo de microcontrolador y la SIE a través de un espacio de memoria conocido con el nombre de memoria RAM USB. Se trata de una memoria de puerto dual especial, la cual se extiende desde el banco de memoria 4 al 7 (400h a 7FFH), teniendo de esta manera un total de 1 Kbyte.

El banco 4 (400h hasta 4FFh) se utiliza específicamente para un control de amortiguación variable, mientras que los bancos del 5 al 7 están disponibles para los datos del USB.

Dependiendo del tipo de búfer que se utiliza, los 8 bytes del banco 4 pueden estar disponibles para ser utilizados como espacio de memoria intermedia USB. Aunque la memoria RAM del USB está disponible para el microcontrolador como memoria de datos, las secciones a las que se accede por la SIE, no deben ser accesibles por el microcontrolador. Para ello se utiliza un mecanismo de “semáforos” para determinar el acceso a un búfer particular en un momento dado.

Interrupciones USB

El módulo USB puede generar múltiples condiciones de interrupción. Para acomodar todas estas fuentes de interrupción, el módulo está provisto de una estructura lógica propia de interrupción, similar a la del microcontrolador. Las interrupciones USB están habilitadas con un conjunto de registros de control y atrapadas en un conjunto independiente del registro de banderas. Todas las fuentes se canalizan en una única solicitud de interrupción USB, USBIF (PIR2 <5>), en la lógica de interrupción del microcontrolador.

Modos de alimentación USB

Muchas aplicaciones USB tendrán diferentes configuraciones y requisitos de alimentación. Los modos de alimentación más comúnmente utilizados son los de Only Power Bus, Self-Power Only y Dual Power con Self-Power Dominance.

Streaming Parallel Port

El Streaming Parallel Port (SPP) constituye una ruta alternativa para los datos. Utilizando el SPP, se puede configurar un punto extremo para enviar o recibir datos directamente desde un hardware externo.

Esta metodología presenta posibilidades de diseño donde el microcontrolador actúa como un administrador de datos, permitiendo al SPP pasar grandes bloques de datos sin la necesidad de que el microcontrolador los procese realmente.

Un ejemplo de aplicación podría incluir un sistema de adquisición de datos, donde los datos se transmiten desde una FIFO externa a través de USB a la computadora host. En este caso, el control del endpoint se gestiona mediante el microcontrolador y el movimiento de datos en bruto se procesa externamente.

Oscilador

El módulo USB posee requisitos específicos de reloj. Para que un equipo funcione a plena velocidad, la frecuencia del reloj debe ser de 48 MHz. Aun así, el núcleo del microcontrolador y otros periféricos no están obligados a correr a dicha velocidad, o incluso utilizar la misma fuente de reloj.

Transferencia Bulk

La transferencia o transmisión Bulk, es una comunicación no periódica, explosiva típicamente empleada por transferencias que requieren usar todo el ancho de banda disponible o en su defecto son demoradas hasta que el ancho de banda completo esté disponible. Esto implica particularmente movimientos de imágenes o video, donde se requiere de gran potencial de transferencia en poco tiempo. Solo se aplica a dispositivos full y high speed.

**Desarrollo del trabajo:**

* **Alimentación del dispositivo:**

Como fuente de alimentación se utiliza un cargador de teléfono celular, que tiene la capacidad de entregar (según sus datos técnicos) 350mA a 9V, valores más que suficientes para las necesidades de todo el proyecto ya que su consumo está rondando los 80mA en las peores condiciones.

En el esquemático pueden observarse las configuraciones circuitales para obtener las tensiones y corrientes necesarias para la alimentación de todo el proyecto. Pero se hará un detalle de los voltajes más significativos para el proyecto.

* +9V: Es la tensión más alta del circuito, es la obtenida directamente desde la fuente de alimentación y de ella se parte para la generación y estabilización del resto de las tensiones necesarias en el resto del circuito. Esquemáticamente se puede observar como entra esta tensión al circuito.
* +5V: A partir de la tensión general de entrada nombrada anteriormente se procede a realizar una regulación de la misma mediante el uso de un LM7805, se puede observar de la siguiente gura las medidas de precaución tomadas tanto para la regulación eficaz de los +5V (agregado de capacitores) así como también las tomadas a fines de proteger la integridad física y el correcto funcionamiento del CI (diodo de protección).
* **Desarrollo del dispositivo:**

La configuración del generador de señales se realiza con interfaz gráfica. Ésta es implementada mediante Python 3.3, incluyendo las librerías de Py.Qt (Versión 4.10.3) y Py.usb (Versión 1.0.0b).

La ventana gráfica de entrada la podemos visualizar a continuación:

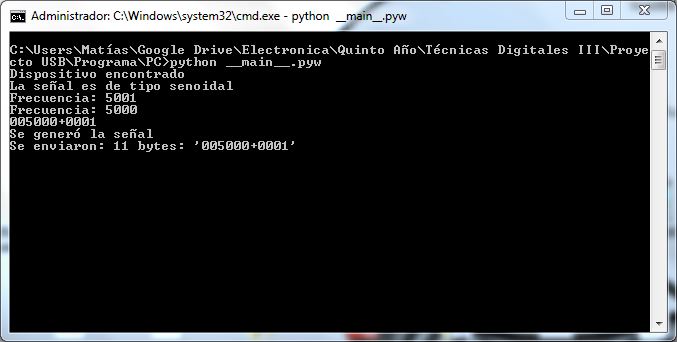


Mediante ésta se puede establecer el tipo de señal a generar: senoidal, triangular, rampa, cuadrada. Además se puede configurar la frecuencia y el offset de dicha señal.

Una vez que se seleccionan los datos, se selecciona el botón “Generar” para realizar dicha función. Además cuenta con las funciones de “Pausa” y “Borrado”.

La interfaz gráfica nos permite la conexión de la PC (Host) al PIC (18F14k50) a través del bus USB.

En la siguiente imagen visualizamos la consola de Python, donde observamos el tipo de señal, la frecuencia de transmisión y el vector de transmisión que se ha generado.



**Código de programa de la interfaz gráfica en Python (Ver adjunto en anexo)**

* **Generación de señales:**

El vector de transmisión generado mediante la interfaz gráfica es transmitido mediante comunicación USB desde la PC al microcontrolador PIC (18F14k50).

El tamaño del vector es de 11 Bytes, los cuales se encuentran codificados en ASCII.

A continuación se adjunta dicho vector con la señal por defecto y se van a explicar cada uno de los bytes que lo componen:

Byte 0: Representa el tipo de señal. (Senoidal=0; Triangular=1, Rampa=2, Cuadrada=3). Byte LSB correspondiente al byte de la izquierda.

Byte 1: es el byte 4 (MSB) de la frecuencia.

Byte 2: es el byte 3 de la frecuencia.

Byte 3: es el byte 2 de la frecuencia.

Byte 4: es el byte 1 de la frecuencia.

Byte 5: es el byte 0 (LSB) de la frecuencia.

Byte 6: Representa el signo del Offset (+ o -).

Byte 7: es el byte 2 del offset.

Byte 8: es el byte 1 del offset.

Byte 9: es el byte 0 del offset.

Byte 10: Representa el byte de control, el cual habilita y deshabilita la generación de la señal. (Inicia=1; Para=0).

Una vez que el PIC (18F14k50) recibe el vector, se encarga de repetirlo para luego enviarlo mediante EUSART al otro PIC (18F4550).

Éste último, utiliza la instrucción de dicho protocolo, para la detección de cada uno de los byte correspondientes. Luego lo que hace es tomar diferentes categorías de byte y asignarle el peso decimal correspondiente para obtener cada uno de los valores determinados.

Tabla de medición de las frecuencias y amplitudes generadas:

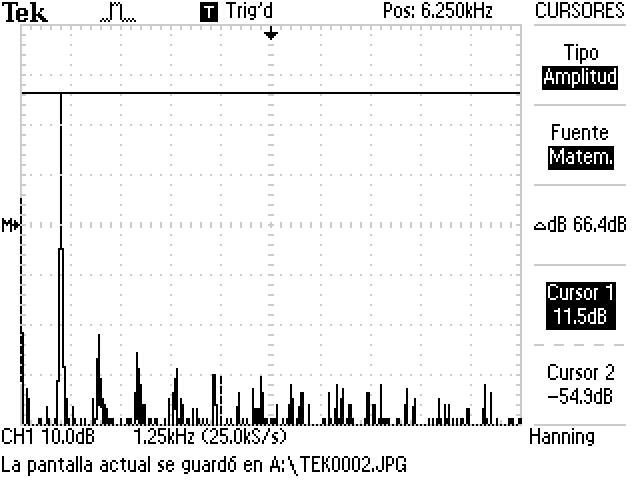
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Frecuencia | Frecuencia  Lograda | Delta [Hz] | Delta [%] | Vo [ V ] | Delta [V] | Delta [%] | Tabla |
| 100 | 100 | 0 | 0 | 4.11 | 0.05 | 1.23 | BAJA FRECUENCIA |
| 200 | 200 | 0 | 0 | 4.10 | 0.04 | 0.98 |
| 300 | 300 | 0 | 0 | 4.10 | 0.04 | 0.98 |
| 400 | 399 | -1 | -0,25 | 4.10 | 0.04 | 0.98 |
| 500 | 502 | 2 | 0,4 | 4.11 | 0.05 | 1.23 |
| 600 | 595 | -5 | -0,8333 | 4.10 | 0.04 | 0.98 |
| 700 | 702 | 2 | 0,2857 | 4.08 | 0.02 | 0.49 |
| 800 | 798 | -2 | -0,25 | 4.1 | 0.04 | 0.985 |
| 900 | 900 | 0 | 0 | 4.1 | 0.04 | 0.985 |
| 1000 | 1003 | 3 | 0,3 | 4.1 | 0.04 | 0.985 |
| 2000 | 2001 | 1 | 0,05 | 4.1 | 0.04 | 0.985 | ALTA FRECUENCIA |
| 3000 | 3010 | 10 | 0,3333 | 4.08 | 0.02 | 0.49 |
| 4000 | 4002 | 2 | 0,05 | 4.06 | 0 | 0 |
| 5000 | 5023 | 23 | 0,46 | 4.04 | -0.02 | -0.49 |
| 6000 | 6010 | 10 | 0,1667 | 4.01 | -0.05 | -1.23 |
| 7000 | 6969 | -31 | -0,4429 | 3.99 | -0.07 | -1.72 |
| 8000 | 8004 | 4 | 0,05 | 3.98 | -0.08 | -1.97 |
| 9000 | 9015 | 15 | 0,1667 | 3.98 | -0.08 | -1.97 |
| 10000 | 10046 | 46 | 0,46 | 3.98 | -0.08 | -1.97 |

Podemos observar en la tabla que las frecuencias logradas se condicen, con un error muy bajo, con las solicitadas.

Para verificar cuan plana es la respuesta en frecuencia del generador se realiza un barrido de frecuencia por todas aquellas que fueron solicitadas. Para esto, se fija la tensión de salida a la máxima solicitada (4 VRMS) para una frecuencia determinada, en este caso de 4 kHz. Luego en cada una de las frecuencias de la tabla se obtiene la tensión de salida. De la pequeña variación de la amplitud de salida respecto del valor esperado, se puede concluir que no se posee una desviación considerable y por ende la respuesta en frecuencia del generador es bastante plana.

Medición de la distorsión armónica en 1 kHz:

Se genera una señal senoidal de máxima amplitud (4 Vrms) y una frecuencia de 1 kHz. A continuación se puede observar una captura del análisis espectral de la señal generada.



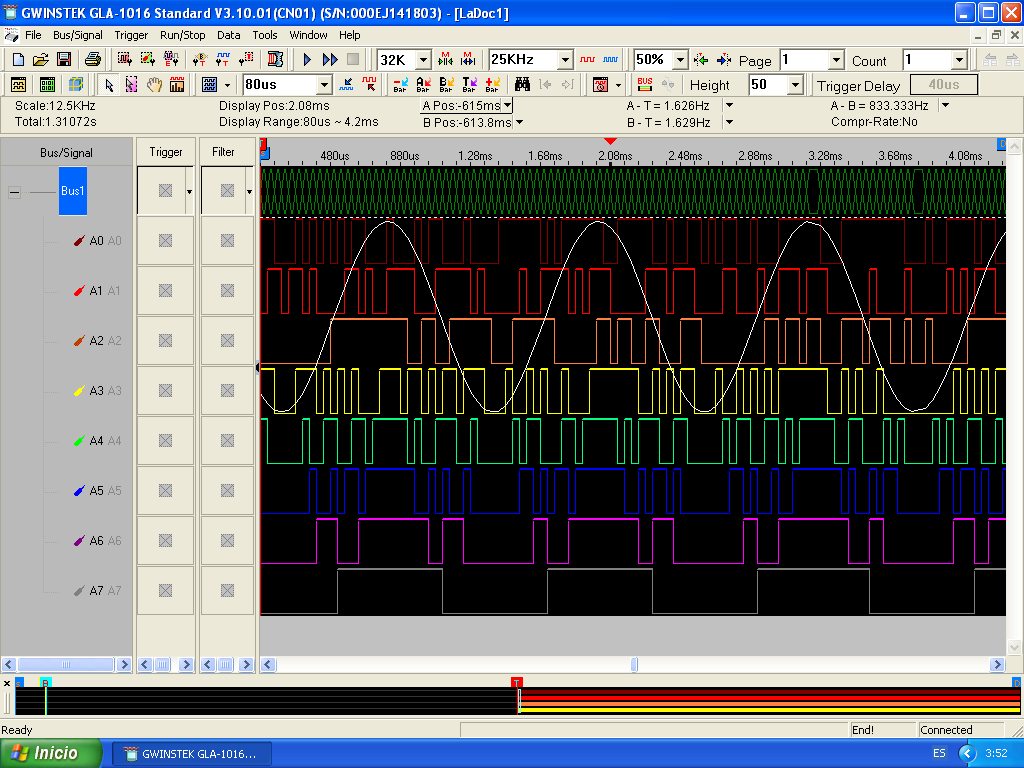
En base a esta imagen se pudieron obtener varias mediciones de amplitud realizada a las armónicas.

A partir de allí se calcula la distorsión armónica total de la señal (THD) como sigue:

Como conclusión podemos destacar que la distorsión armónica total de la señal, es muy insignificante (< al 1%), con lo cual obtenemos una muy buena pureza de la señal.

Medición de bus de datos con analizador lógico para corroborar secuencia binaria:

En la siguiente medición se utiliza un analizador lógico. A él ingresan los ocho bits del bus de datos del puerto D del PIC (18F4550) en paralelo, debido que éstos se dirigen al conversor D/A. Entonces, mediante las conmutaciones realizadas por parte del PIC se puede observar la secuencia binaria seguida para generar la señal senoidal (en base a tablas de datos programadas), lo cual se observa en la siguiente imagen:



Nota: Vale destacar que las conmutaciones observadas fueron realizadas para una frecuencia de señal senoidal de 1 kHz y que se tomó una frecuencia de muestreo de 25 kHz y una memoria de 32 K.

Finalmente, si al bus de datos (ocho bits del puerto), se los pondera en binario natural y se los suma, obtenemos un número que representa una magnitud (en éste caso es una tensión) de la señal en ese punto, obteniendo la señal senoidal propiamente dicha.

**Código de programa del PIC 18F4550 (Ver adjunto en anexo)**

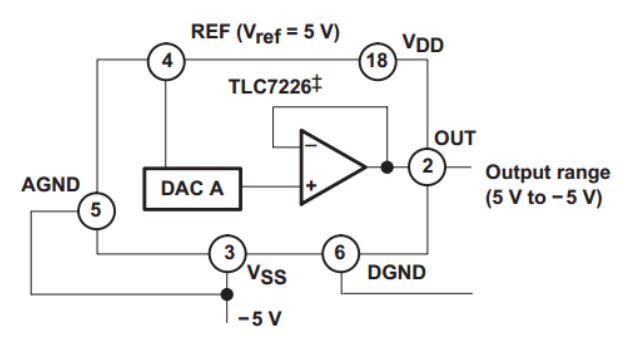
**Código de programa del PIC 18F14k50 (Ver adjunto en anexo)**

Finalmente con los valores finales obtenidos en el PIC (18F4550) se transmiten por el puerto D hacia un conversor D/A para generar la señal configurada.

Conversor Digital-Analógico:

El conversor digital-analógico utilizado es fabricado por la empresa Texas Instruments, específicamente se trata del circuito integrado TLC7226, el cual cuenta con tiempos de conversión bajos debido a que cuenta con carga en paralelo del byte a convertir. Tiene cuatro salidas analógicas diferentes disponibles (todas demultiplexas desde las mismas 8 entradas) de las cuales solo se utiliza una para este proyecto. Este también es capaz de entregar hasta 5 mA de corriente de salida, por lo que según la aplicación deberá analizarse si se necesita o no, una etapa capaz de entregar una mayor corriente.

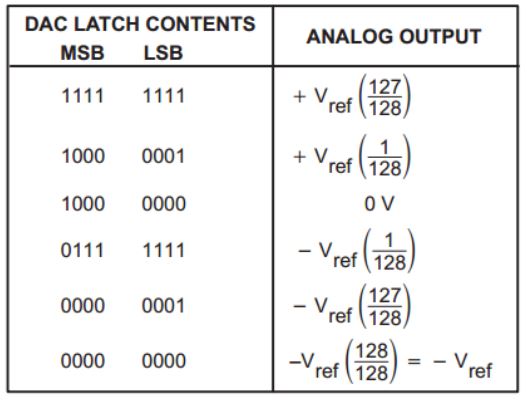
La forma de conexión utilizada es una de las propuestas en la hoja de datos del fabricante, en ella es posible lograr una tensión equivalente de salida entre Vref. A continuación puede observarse la configuración planteada:



Teniendo en cuenta que por la disposición circuital utilizada en el DAC se pueden lograr señales en el rango de los 5V, y que el valor a representar es siempre de 1 byte (cada valor se representa por 8 bits cada 8 bits) y que ese es el ancho del bus del DAC, se tiene un total de 28 = 256 valores posibles diferentes lo que generará un salto o delta de tensión mínimo de:

Que será el mínimo escalón que podrá ser logrado por el DAC para la representación entre dos números binarios seguidos de entrada.

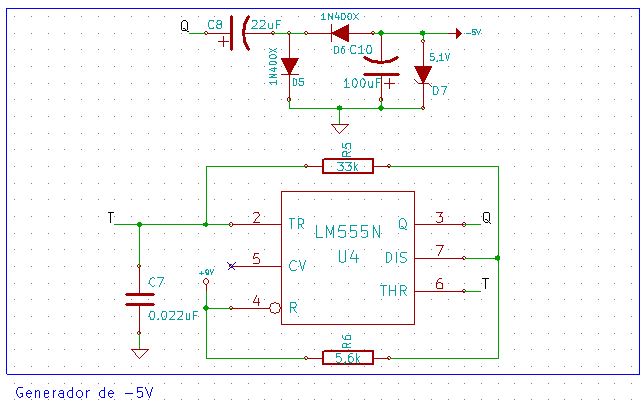
De la hoja de datos del DAC para la citada configuración se observa la forma de interpretación de los datos enviados desde el µC por el DAC para lograr cada valor de tensión:



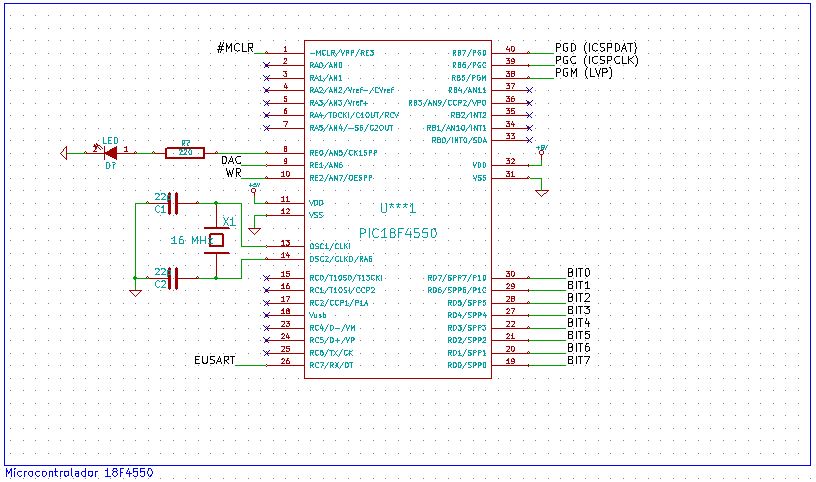
Como se puede verificar de la tabla, si quiere lograrse el máximo de tensión positiva (+5) debe ser colocado en la entrada del DAC el valor 0 x FF lo que dará una tensión analógica de salida del 99,21% de , es decir . Para lograrse una salida nula, esto es 0 V el valor a enviar es 0 x 80 y por ende cualquier valor por debajo de este provocará una salida de tensión negativa.

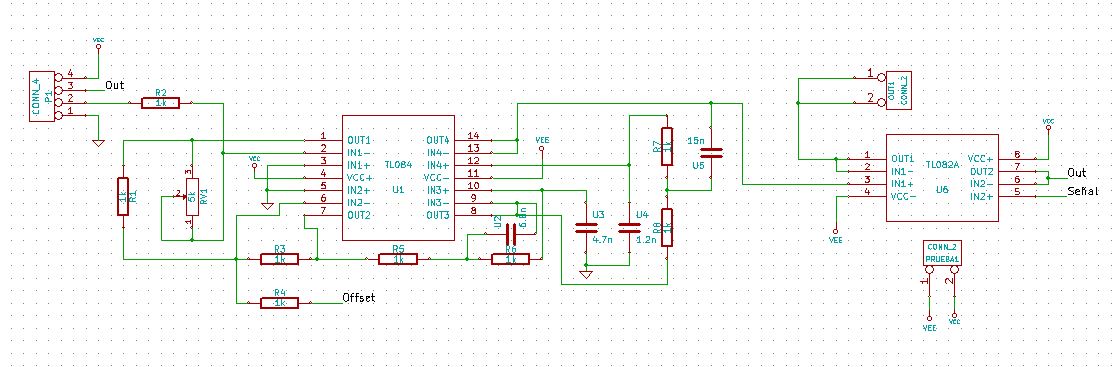
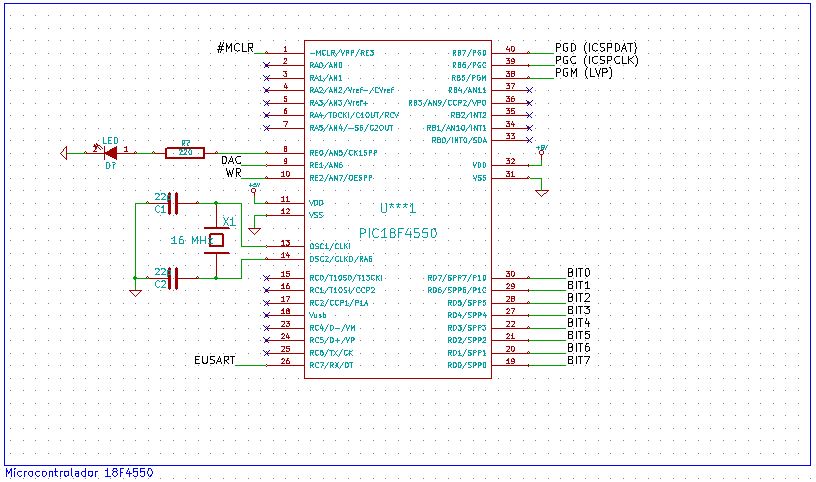
Si se desea alcanzar el máximo de tensión negativa (-5V) debe enviarse el valor 0 x 00 para lo que se logrará una tensión analógica de salida del 100 % de , o sea .

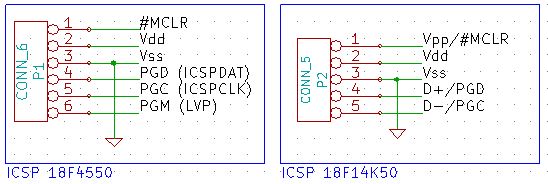
* **Diagramas Circuitales:**
* **Generador de -5V**



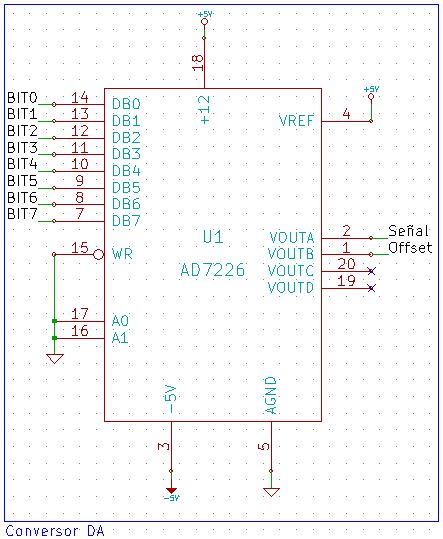
* **Configuración del PIC 18F4550**



* **Filtro, sumador y seguidor de emisor**
* **Configuración del PIC 18F14K50**
* **ICSP De ambos Microcontroladores**



* **Conversor Digital/Analógico (DAC)**

****

**Anexo:**

**Código de programa de la interfaz gráfica en Python:**

from MainWindow import Ui\_MainWindow #Se importa la configuracion de la ventana principal

from Configuracion import Ui\_ConfWindow #Se importa la ventana de configuracion

from AcercaDe import Ui\_AcercaWindow #Se importa la ventana de informacion

from PyQt4 import QtCore, QtGui #Se importa la libreria de GUI

from PyQt4.QtCore import SIGNAL, QObject

import sys #Se importa la libreria del sistema

import array #Se importa la libreria de manejo de cadenas

import os #Se importa la libreria del SO

import usb.core #Se importa la libreria del BUS USB del sistema

import usb.util

import time #Se importa la libreria de tiempo del sistema

try:

\_fromUtf8 = QtCore.QString.fromUtf8

except AttributeError:

def \_fromUtf8(s):

return s

class MainWindow(QtGui.QMainWindow): #Configuracion basica, de SEÑALES y de SLOTS

de la GUI (ventana principal)

def \_\_init\_\_(self, parent=None):

super(MainWindow, self).\_\_init\_\_(parent)

QtGui.QWidget.\_\_init\_\_(self, parent)

self.ui = Ui\_MainWindow()

self.ui.setupUi(self)

self.ui.actionSalir.triggered.connect(self.close)

self.ui.actionConfiguracion.triggered.connect(self.Configuracion)

self.ui.actionAcerca\_de.triggered.connect(self.Acerca)

self.ui.generar.clicked.connect(self.Generar)

self.ui.pausar.clicked.connect(self.Pausar)

self.ui.borrar.clicked.connect(self.Borrar)

self.ui.borrar.clicked.connect(self.ui.senoidal.click)

self.ui.offset.valueChanged.connect(self.Off)

self.ui.frecuencia.valueChanged.connect(self.Frec)

self.ui.senoidal.clicked.connect(self.Senoidal)

self.ui.triangular.clicked.connect(self.Triangular)

self.ui.rampa.clicked.connect(self.Rampa)

self.ui.cuadrada.clicked.connect(self.Cuadrada)

def Generar(self): #Funcion del boton "Generar"

global Control

Control = '1'

try:

Config = self.Configurar()

signal = dev.write(1, Config, 0, 100)

if signal:

print('Se generó la señal')

print('Se enviaron: ' + str(signal) + ' bytes: \'' + str(Config) + '\'')

print('')

else:

print('Se produjo un error en el envio, revise la configuración')

except:

print ('No se pudo configurar el dispositivo')

def Pausar(self): #Funcion del boton "Pausar"

global Control

global Config

if Control == '1' and Control != '0':

Control = '0'

Config = self.Configurar()

signal = dev.write(1, Config, 0, 100)

print('Se pausó la señal')

print('Se enviaron: ' + str(signal) + ' bytes: \'' + str(Config) + '\'')

print('')

else:

print('La señal ya se encuentra pausada')

def Borrar(self): #Funcion del boton "Borrar"

global Tipo

global Frecuencia

global Offset

global Control

global Config

Tipo = '0' #Seno (tipo por defecto)

Frecuencia = '5000' #5000 Hz (frecuencia por defecto)

Offset = '0' #0V de offset (offset por defecto)

Control = '0' #Desactivo la generación de la señal

self.ui.frecuencia.setValue(5000)

self.ui.offset.setValue(0)

Config = self.Configurar()

try:

signal = dev.write(1, Config, 0, 100)

if signal:

print('Se configuró la señal por defecto')

print('Se enviaron: ' + str(signal) + ' bytes: \'' + str(Config) + '\'')

print('')

else:

print('Se produjo un error en el envio, revise la configuración')

except:

print ('No se pudo configurar el dispositivo')

def Senoidal(self): #Funcion del radio boton "Senoidal"

global Tipo

Tipo = '0'

print('La señal es de tipo senoidal')

def Triangular(self): #Funcion del radio boton "Triangular"

global Tipo

Tipo = '1'

print('La señal es de tipo triangular')

def Rampa(self): #Funcion del radio boton "Rampa"

global Tipo

Tipo = '2'

print('La señal es de tipo rampa')

def Cuadrada(self): #Funcion del radio boton "Cuadrada"

global Tipo

Tipo = '3'

print('La señal es de tipo cuadrada')

def Frec(self, Frec): #Funcion del spinbox de "Frecuencia"

global Frecuencia

Frecuencia = str(Frec)

print('Frecuencia: ' + Frecuencia)

def Off (self, Off): #Funcion del slider de "Offset"

global Offset

self.ui.offset.setToolTip(\_fromUtf8(str(-self.ui.offset.value())))

self.ui.lOffset.setText(\_fromUtf8(str(-self.ui.offset.value()) + ' %'))

Offset = str(-Off)

print('Offset: ' + Offset)

def Configurar(self): #Funcion que genera el vector de

global Tipo # configuracion de la señal a enviar por USB

global Frecuencia

global Offset

global Control

"""

ORDEN DE LOS BYTES A ENVIAR:

1er BYTE --> TIPO DE SEÑAL [0 SENOIDAL,

1 TRIANGULAR,

2 RAMPA,

3 CUADRADA] --> N° en ASCII

2do BYTE --> FRECUENCIA [4] --------------> N° en ASCII

3er BYTE --> FRECUENCIA [3] --------------> N° en ASCII

4to BYTE --> FRECUENCIA [2] --------------> N° en ASCII

5to BYTE --> FRECUENCIA [1] --------------> N° en ASCII

6to BYTE --> FRECUENCIA [0] --------------> N° en ASCII

7mo BYTE --> OFFSET [SIGNO] --------------> Signo (+ o -) en ASCII

8vo BYTE --> OFFSET [2] ------------------> N° en ASCII

9no BYTE --> OFFSET [1] ------------------> N° en ASCII

10mo BYTE --> OFFSET [0] -----------------> N° en ASCII

11vo BYTE --> CONTROL [0 PARA, 1 INICIA] -> N° en ASCII

"""

if len(str(Frecuencia)) < 5 and len(str(Frecuencia)) > 3:

Frecuencia = '0' + Frecuencia

elif len(str(Frecuencia)) < 4 and len(str(Frecuencia)) > 2:

Frecuencia = '00' + Frecuencia

elif len(str(Frecuencia)) < 3 and len(str(Frecuencia)) > 1:

Frecuencia = '000' + Frecuencia

elif len(str(Frecuencia)) < 2:

Frecuencia = '0000' + Frecuencia

if int(Offset) >= 0:

if len(Offset) == 3:

Offset = '+' + Offset

elif len(Offset) < 3 and len(Offset) > 1:

Offset = '+0' + Offset

elif len(Offset) < 2:

Offset = '+00' + Offset

elif int(Offset) < 0:

if len(Offset) < 4 and len(Offset) > 2:

Offset = '-0' + str(-int(Offset))

elif len(Offset) < 3:

Offset = '-00' + str(-int(Offset))

Config = Tipo + str(Frecuencia) + str(Offset) + Control

print(Config)

return Config

def Configuracion(self): #Funcion de llamada a la ventana de "Configuracion"

self.w = ConfWindow()

#self.w.show()

self.w.exec\_() #Una vez abierta solo se puede interactuar con esta ventana

def Acerca(self): #Funcion de llamada a la ventana de informacion "Acerca de..."

self.w = AcercaWindow()

#self.w.show()

self.w.exec\_() #Una vez abierta solo se puede interactuar con esta ventana

class ConfWindow(QtGui.QDialog): #Configuracion basica, de SEÑALES y de SLOTS de la #GUI (ventana de configuracion)

def \_\_init\_\_(self, parent=None):

super(ConfWindow, self).\_\_init\_\_(parent)

QtGui.QWidget.\_\_init\_\_(self, parent)

self.ui = Ui\_ConfWindow()

self.ui.setupUi(self)

class AcercaWindow(QtGui.QDialog): #Configuracion basica, de SEÑALES y de SLOTS de

def \_\_init\_\_(self, parent=None): # la GUI (ventana de informacion)

super(AcercaWindow, self).\_\_init\_\_(parent)

QtGui.QWidget.\_\_init\_\_(self, parent)

self.ui = Ui\_AcercaWindow()

self.ui.setupUi(self)

self.ui.bSalir.clicked.connect(self.close)

#VALORES INICIALES DE LA SEÑAL

Tipo = '0'

Frecuencia = '5000'

Offset = '0'

Control = '1'

try: #Busqueda del dispositivo por VID y PID

dev = usb.core.find(idVendor=0x04D8, idProduct=0x000B)

print('Dispositivo encontrado')

except:

print('Dispositivo no encontrado')

if dev is None:

MainWindow().ui.setStatusTip('Dispositivo no encontrado')

raise ValueError('Dispositivo no encontrado')

else:

dev.set\_configuration()

cfg = dev.get\_active\_configuration()

cfg = dev[0]

intf = cfg[(0,0)]

ep = usb.util.find\_descriptor(

intf,

#Se asocia al primer ENDPOINT de salida

custom\_match = \

lambda e: \

usb.util.endpoint\_direction(e.bEndpointAddress) == \

usb.util.ENDPOINT\_OUT

)

ep = intf[0]

interface = 0

endpoint = dev[0][(0,0)][0]

assert ep is not None

if \_\_name\_\_ == '\_\_main\_\_': #Se inicia el bucle de ejecucion del programa

app = QtGui.QApplication(sys.argv)

myapp = MainWindow()

myapp.show()

sys.exit(app.exec\_())

**Código de programa del PIC 18F4550:**

#INCLUDE <18F4550.h>

#FUSES HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL4,CPUDIV1,VREGEN

#FUSES BROWNOUT\_NOSL, NOBROWNOUT, PUT, NOCPD, NOSTVREN

#FUSES NOWRT, NOWRTD, IESO, NOFCMEN, NOPBADEN, NOWRTC, NOWRTB

#FUSES NOEBTR, NOEBTRB, NOCPB, MCLR, NOLPT1OSC, NOXINST

#FUSES NOICPRT, BORV45

#USE delay(clock=48000000)

#USE fast\_io(d)

#USE rs232(baud=57600,parity=N,xmit=PIN\_C6,rcv=PIN\_C7,bits=8, ERRORS)

#INCLUDE <string.h>

#INCLUDE <stdlib.h>

#INCLUDE <math.h>

#INCLUDE <stdlib.h>

#DEFINE LED PIN\_E0

#DEFINE DAC\_B PIN\_E1

#DEFINE WR PIN\_E2

#DEFINE Enciende Output\_High

#DEFINE Apaga Output\_Low

#DEFINE Conmuta Output\_Toggle

#DEFINE Tipo recbuf[0]

#DEFINE Frec\_4 recbuf[1]

#DEFINE Frec\_3 recbuf[2]

#DEFINE Frec\_2 recbuf[3]

#DEFINE Frec\_1 recbuf[4]

#DEFINE Frec\_0 recbuf[5]

#DEFINE Offset\_3 recbuf[6]

#DEFINE Offset\_2 recbuf[7]

#DEFINE Offset\_1 recbuf[8]

#DEFINE Offset\_0 recbuf[9]

#DEFINE Control recbuf[10]

#DEFINE DOSPI 2\*pi

#BYTE FSR0L = 0XFE9

#BYTE FSR0H = 0XFEA

#BYTE INDF0 = 0XFEF

#BYTE PORTB = 0XF81

#BYTE PORTD = 0XF83

#BYTE STATUS = 0XFD8

#BYTE RCREG = 0XFAE

const int8 Lenbuf = 12;

int i = 0;

int j;

char recbuf[Lenbuf] = {'0','0','0','0','0','0','0','0','0','0','0'};

float temp;

long Frec = 0;

int Off = 0;

int estado = 0;

unsigned int8 SENO\_ALTA[256] = {128,131,134,137,140,143,146,149,153,156,159,162,

165,168,171,174,177,180,182,185,188,191,194,196,199,201,204,207,209,211,214,216,

218,220,223,225,227,229,231,232,234,236,238,239,241,242,243,245,246,247,248,249,

250,251,252,253,253,254,254,255,255,255,255,255,255,255,255,255,255,254,254,253,

253,252,251,251,250,249,248,247,245,244,243,241,240,238,237,235,233,232,230,228,

226,224,222,219,217,215,213,210,208,205,203,200,198,195,192,189,187,184,181,178,

175,172,169,166,163,160,157,154,151,148,145,142,139,135,132,129,126,123,120,116,

113,110,107,104,101,98,95,92,89,86,83,80,77,74,71,68,66,63,60,57,55,52,50,47,45,

42,40,38,36,33,31,29,27,25,23,22,20,18,17,15,14,12,11,10,8,7,6,5,4,4,3,2,2,1,1,

0,0,0,0,0,0,0,0,0,0,1,1,2,2,3,4,5,6,7,8,9,10,12,13,14,16,17,19,21,23,24,26,28,

30,32,35,37,39,41,44,46,48,51,54,56,59,61,64,67,70,73,75,78,81,84,87,90,93,96,

99,102,106,109,112,115,118,121,124,127};

unsigned int8 SENO\_BAJA[1024] = {128,128,129,130,131,131,132,133,134,135,135,136,

137,138,138,139,140,141,142,142,143,144,145,146,146,147,148,149,149,150,151,152,

152,153,154,155,156,156,157,158,159,159,160,161,162,162,163,164,165,165,166,167,

168,168,169,170,171,171,172,173,174,174,175,176,177,177,178,179,179,180,181,182,

182,183,184,184,185,186,187,187,188,189,189,190,191,191,192,193,193,194,195,195,

196,197,197,198,199,199,200,201,201,202,203,203,204,204,205,206,206,207,208,208,

209,209,210,211,211,212,212,213,214,214,215,215,216,216,217,218,218,219,219,220,

220,221,221,222,222,223,223,224,224,225,226,226,227,227,228,228,228,229,229,230,

230,231,231,232,232,233,233,234,234,234,235,235,236,236,237,237,237,238,238,239,

239,239,240,240,240,241,241,242,242,242,243,243,243,244,244,244,245,245,245,246,

246,246,246,247,247,247,248,248,248,248,249,249,249,249,250,250,250,250,250,251,

251,251,251,252,252,252,252,252,252,253,253,253,253,253,253,254,254,254,254,254,

254,254,254,254,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,

255,255,255,255,255,254,254,254,254,254,254,254,254,253,253,253,253,253,253,253,

252,252,252,252,252,251,251,251,251,251,250,250,250,250,249,249,249,249,248,248,

248,248,247,247,247,247,246,246,246,245,245,245,244,244,244,243,243,243,242,242,

242,241,241,241,240,240,240,239,239,238,238,238,237,237,236,236,235,235,235,234,

234,233,233,232,232,232,231,231,230,230,229,229,228,228,227,227,226,226,225,225,

224,224,223,223,222,222,221,221,220,219,219,218,218,217,217,216,216,215,214,214,

213,213,212,211,211,210,210,209,208,208,207,207,206,205,205,204,203,203,202,202,

201,200,200,199,198,198,197,196,196,195,194,194,193,192,192,191,190,190,189,188,

188,187,186,185,185,184,183,183,182,181,180,180,179,178,178,177,176,175,175,174,

173,173,172,171,170,170,169,168,167,167,166,165,164,164,163,162,161,161,160,159,

158,157,157,156,155,154,154,153,152,151,151,150,149,148,147,147,146,145,144,144,

143,142,141,140,140,139,138,137,137,136,135,134,133,133,132,131,130,129,129,128,

127,126,126,125,124,123,122,122,121,120,119,118,118,117,116,115,115,114,113,112,

111,111,110,109,108,108,107,106,105,104,104,103,102,101,101,100,99,98,98,97,96,

95,94,94,93,92,91,91,90,89,88,88,87,86,85,85,84,83,82,82,81,80,80,79,78,77,77,76,

75,75,74,73,72,72,71,70,70,69,68,67,67,66,65,65,64,63,63,62,61,61,60,59,59,58,57,

57,56,55,55,54,53,53,52,52,51,50,50,49,48,48,47,47,46,45,45,44,44,43,42,42,41,41,

40,39,39,38,38,37,37,36,36,35,34,34,33,33,32,32,31,31,30,30,29,29,28,28,27,27,26,

26,25,25,24,24,23,23,23,22,22,21,21,20,20,20,19,19,18,18,17,17,17,16,16,15,15,15,

14,14,14,13,13,13,12,12,12,11,11,11,10,10,10,9,9,9,8,8,8,8,7,7,7,7,6,6,6,6,5,5,5,

5,4,4,4,4,4,3,3,3,3,3,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,2,2,2,2,

2,2,3,3,3,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,7,7,7,7,8,8,8,9,9,9,9,10,10,10,11,11,

11,12,12,12,13,13,13,14,14,15,15,15,16,16,16,17,17,18,18,18,19,19,20,20,21,21,21,

22,22,23,23,24,24,25,25,26,26,27,27,27,28,28,29,29,30,31,31,32,32,33,33,34,34,35,

35,36,36,37,37,38,39,39,40,40,41,41,42,43,43,44,44,45,46,46,47,47,48,49,49,50,51,

51,52,52,53,54,54,55,56,56,57,58,58,59,60,60,61,62,62,63,64,64,65,66,66,67,68,68,

69,70,71,71,72,73,73,74,75,76,76,77,78,78,79,80,81,81,82,83,84,84,85,86,87,87,88,

89,90,90,91,92,93,93,94,95,96,96,97,98,99,99,100,101,102,103,103,104,105,106,106,

107,108,109,109,110,111,112,113,113,114,115,116,117,117,118,119,120,120,121,122,

123,124,124,125,126,127,127};

#BYTE SENO\_ALTA = 0x100

#BYTE SENO\_BAJA = 0x200

void generar();

#INT\_RDA //Manejo de la interrupcion de EUSART

void RDA\_ISR(){

char letra = 0x00;

estado = 0;

letra = getchar();

recbuf[i] = letra;

i++;

if(i==11){

estado = 1;

i=0;

}

}

void main(void)

{

clear\_interrupt (INT\_RDA);

enable\_interrupts (INT\_RDA);

enable\_interrupts (GLOBAL);

set\_tris\_a(0b00000000);

set\_tris\_b(0b00000000);

set\_tris\_d(0b00000000);

set\_tris\_e(0b00000000);

delay\_ms(10);

while (TRUE){

if(Control=='1'){ //Se puede generar la señal

for(j=0;j<=4;j++){ //FRECUENCIA

switch(recbuf[j+1]){

case '0':

temp = 0;

break;

case '1':

temp = 1;

break;

case '2':

temp = 2;

break;

case '3':

temp = 3;

break;

case '4':

temp = 4;

break;

case '5':

temp = 5;

break;

case '6':

temp = 6;

break;

case '7':

temp = 7;

break;

case '8':

temp = 8;

break;

case '9':

temp = 9;

break;

}

if(j==0){ //MSB de frecuencia

Frec = temp \* 10000;

}

if(j==1){

Frec = Frec + temp \* 1000;

}

if(j==2){

Frec = Frec + temp \* 100;

}

if(j==3){

Frec = Frec + temp \* 10;

}

if(j==4){ //LSB de frecuencia

Frec = Frec + temp;

}

}

for(j=0;j<=2;j++){ //OFFSET

switch(recbuf[j+7]){

case '0':

temp = 0;

break;

case '1':

temp = 1;

break;

case '2':

temp = 2;

break;

case '3':

temp = 3;

break;

case '4':

temp = 4;

break;

case '5':

temp = 5;

break;

case '6':

temp = 6;

break;

case '7':

temp = 7;

break;

case '8':

temp = 8;

break;

case '9':

temp = 9;

break;

}

if(j==0){ //MSB del Offset

Off = temp \* 100;

}

if(j==1){

Off = Off + temp \* 10;

}

if(j==2){ //LSB del Offset

Off = Off + temp;

}

}

if(Offset\_3 == '+'){

Off = ceil(Off\*127/100) + 128;

Enciende(DAC\_B); //Selecciono el DAC B

delay\_ms(2);

Apaga(WR); //WR debe estar bajo para cambiar de DAC

output\_d(Off); //Offset DAC

delay\_ms(1);

Enciende(WR); //Latch al DAC B (Offset)

delay\_ms(1);

Apaga(WR); //WR debe estar bajo para cambiar de DAC

Apaga(DAC\_B); //Apago el DAC B (selecciono el DAC A - Señal)

Apaga(WR);

}

if(Offset\_3 == '-'){

Off = ceil(-Off\*128/100) + 128;

Enciende(DAC\_B); //Selecciono el DAC B

delay\_ms(2);

Apaga(WR); //WR debe estar bajo para cambiar de DAC

output\_d(Off); //Offset DAC

delay\_ms(1);

Enciende(WR); //Latch al DAC B (Offset)

delay\_ms(1);

Apaga(WR); //WR debe estar bajo para cambiar de DAC

Apaga(DAC\_B); //Apago el DAC B (selecciono el DAC A - Señal)

Apaga(WR);

}

if (estado==1){

generar();

}

}

if(Control=='0'){ // Control es 0 y no se puede generar

Apaga(WR); //WR debe estar bajo para cambiar de DAC

Enciende(DAC\_B); //Selecciono el DAC B

output\_d(0x80); //Offset DAC

Enciende(WR); //Latch al DAC B (Offset)

Apaga(WR); //WR debe estar bajo para cambiar de DAC

Apaga(DAC\_B); //Apago el DAC B (selecciono el DAC A - Señal)

output\_d (0x80); // 0V DAC

}

}

}

void generar(){

clear\_interrupt(INT\_RDA);

enable\_interrupts(INT\_RDA);

switch(Tipo){

case '0': //Señal senoidal

Enciende(LED);

delay\_ms(200);

Apaga(LED);

switch(Frec){

case 0: //0 Hz

#ASM ASIS

RETORNO\_0:

MOVLW 128

MOVWF PORTD

GOTO RETORNO\_0

#ENDASM

break;

case 100: //100 Hz

#ASM ASIS

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_100:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_1\_100

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_1\_100

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_1\_100:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_2\_100

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_2\_100

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_2\_100:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_3\_100

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_3\_100

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_3\_100:

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

GOTO RETORNO\_100

#ENDASM

break;

case 200: //200 Hz

#ASM ASIS

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_200:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_1\_200

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_1\_200

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_1\_200:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_2\_200

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_2\_200

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_2\_200:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_3\_200

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_3\_200

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_3\_200:

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

GOTO RETORNO\_200

#ENDASM

break;

case 300: //300 Hz

#ASM ASIS

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_300:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_1\_300

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_1\_300

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_1\_300:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_2\_300

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_2\_300

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_2\_300:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_3\_300

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_3\_300

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_3\_300:

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP

GOTO RETORNO\_300

#ENDASM

break;

case 400: //400 Hz

#ASM ASIS

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_400:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_1\_400

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_1\_400

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_1\_400:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_2\_400

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_2\_400

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_2\_400:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_3\_400

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_3\_400

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_3\_400:

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP

GOTO RETORNO\_400

#ENDASM

break;

case 500: //500 Hz

#ASM ASIS

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_500:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_1\_500

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_1\_500

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_1\_500:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_2\_500

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_2\_500

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_2\_500:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_3\_500

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_3\_500

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_3\_500:

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP

GOTO RETORNO\_500

#ENDASM

break;

case 600: //600 Hz

#ASM ASIS

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_600:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_1\_600

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_1\_600

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_1\_600:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_2\_600

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_2\_600

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_2\_600:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_3\_600

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_3\_600

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_3\_600:

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

GOTO RETORNO\_600

#ENDASM

break;

case 700: //700 Hz

#ASM ASIS

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_700:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_1\_700

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_1\_700

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_1\_700:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_2\_700

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_2\_700

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_2\_700:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_3\_700

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_3\_700

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_3\_700:

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

GOTO RETORNO\_700

#ENDASM

break;

case 800: //800 Hz

#ASM ASIS

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_800:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_1\_800

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_1\_800

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_1\_800:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_2\_800

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_2\_800

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_2\_800:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_3\_800

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_3\_800

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_3\_800:

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

GOTO RETORNO\_800

#ENDASM

break;

case 900: //900 Hz

#ASM ASIS

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_900:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_1\_900

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_1\_900

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_1\_900:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_2\_900

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_2\_900

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_2\_900:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_3\_900

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_3\_900

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_3\_900:

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP

GOTO RETORNO\_900

#ENDASM

break;

case 1000: //1000 Hz

#ASM ASIS

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_1000:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_1\_1000

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_1\_1000

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_1\_1000:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_2\_1000

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_2\_1000

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_2\_1000:

MOVF INDF0, W

MOVWF PORTD

INCFSZ FSR0L, 1

GOTO SALTO\_3\_1000

INCF FSR0H, 1

MOVLW 6

SUBWF FSR0H, 0

BTFSS STATUS, 2

GOTO SALTO\_3\_1000

MOVLW 2

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

SALTO\_3\_1000:

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

GOTO RETORNO\_1000

#ENDASM

break;

case 2000: //2000 Hz

#ASM ASIS

MOVLW 1

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_2000:

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

GOTO RETORNO\_2000

#ENDASM

break;

case 3000: //3000 Hz

#ASM ASIS

MOVLW 1

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_3000:

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP

GOTO RETORNO\_3000

#ENDASM

break;

case 4000: //4000 Hz

#ASM ASIS

MOVLW 1

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_4000:

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP

GOTO RETORNO\_4000

#ENDASM

break;

case 5000: //5000 Hz

#ASM ASIS

MOVLW 1

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_5000:

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

GOTO RETORNO\_5000

#ENDASM

break;

case 6000: //6000 Hz

#ASM ASIS

MOVLW 1

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_6000:

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP

GOTO RETORNO\_6000

#ENDASM

break;

case 7000: //7000 Hz

#ASM ASIS

MOVLW 1

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_7000:

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

GOTO RETORNO\_7000

#ENDASM

break;

case 8000: //8000 Hz

#ASM ASIS

MOVLW 1

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_8000:

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

GOTO RETORNO\_8000

#ENDASM

break;

case 9000: //9000 Hz

#ASM ASIS

MOVLW 1

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_9000:

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

NOP NOP NOP

NOP NOP NOP

NOP NOP NOP

GOTO RETORNO\_9000

#ENDASM

break;

case 10000: //10000 Hz

#ASM ASIS

MOVLW 1

MOVWF FSR0H

MOVLW 0

MOVWF FSR0L

RETORNO\_10000:

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

MOVF INDF0, W

MOVWF PORTD

INCF FSR0L

NOP NOP NOP

GOTO RETORNO\_10000

#ENDASM

break;

}

break;

case '1': //Señal triangular

Enciende(LED);

delay\_ms(200);

Apaga(LED);

delay\_ms(200);

Enciende(LED);

delay\_ms(200);

Apaga(LED);

delay\_ms(500);

while(TRUE){

for(j=0;j<=255;j++){

output\_d(j);

delay\_us(1);

}

for(j=255;j>=0;j--){

output\_d(j);

delay\_us(1);

}

}

break;

case '2': //Señal rampa

Enciende(LED);

delay\_ms(200);

Apaga(LED);

delay\_ms(200);

Enciende(LED);

delay\_ms(200);

Apaga(LED);

delay\_ms(200);

Enciende(LED);

delay\_ms(200);

Apaga(LED);

delay\_ms(500);

while(TRUE){

for(j=0;j<=255;j++){

output\_d(j);

delay\_us(1);

}

}

break;

case '3': //Señal cuadrada

Enciende(LED);

delay\_ms(200);

Apaga(LED);

delay\_ms(200);

Enciende(LED);

delay\_ms(200);

Apaga(LED);

delay\_ms(200);

Enciende(LED);

delay\_ms(200);

Apaga(LED);

delay\_ms(200);

Enciende(LED);

delay\_ms(200);

Apaga(LED);

delay\_ms(500);

while(TRUE){

output\_d(0x00);

delay\_us(7);

output\_d(0xFF);

delay\_us(7);

}

break;

}

}

**Código de programa del PIC 18F14k50:**

#INCLUDE <18F14K50.h>

#FUSES HS

#FUSES PLLEN

#FUSES PCLKEN

#FUSES CPUDIV1

#FUSES USBDIV1

#FUSES NOFCMEN

#FUSES NOWDT

#FUSES WDT128

#FUSES NOIESO

#FUSES MCLR

#FUSES NOLVP

#FUSES NOXINST

#FUSES NODEBUG

#FUSES PUT

#FUSES NODEBUG

#FUSES NOBROWNOUT

#FUSES NOPROTECT

#USE delay(clock=48000000)

#USE rs232(baud=57600, parity=N, xmit=PIN\_B7, rcv=PIN\_B5 ,bits=8, ERRORS)

#DEFINE USB\_HID\_DEVICE FALSE // Deshabilitamos el uso de las directivas HID

#DEFINE USB\_EP1\_TX\_ENABLE USB\_ENABLE\_BULK // Turn on EP1(EndPoint1) for IN

// bulk/interrupt transfers

#DEFINE USB\_EP1\_RX\_ENABLE USB\_ENABLE\_BULK // Turn on EP1(EndPoint1) for

// OUT bulk/interrupt transfers

#DEFINE USB\_EP1\_TX\_SIZE 32 // Size to allocate for the tx endpoint 1 buffer

#DEFINE USB\_EP1\_RX\_SIZE 32 // Size to allocate for the rx endpoint 1 buffer

#INCLUDE <pic18\_usb.h> // Microchip PIC18Fxx5x Hardware layer for CCS's PIC

// USB driver

#INCLUDE <rr2\_USB\_Monitor.h> // Configuración del USB y los descriptores para este

// dispositivo

#INCLUDE <usb.c> // Handles usb setup tokens and get descriptor reports

#INCLUDE <math.h>

#INCLUDE <stdlib.h>

#DEFINE Enciende Output\_High

#DEFINE Apaga Output\_Low

#DEFINE Conmuta Output\_Toggle

#DEFINE LED\_ROJO PIN\_C0 //Conectado

#DEFINE LED\_VERDE PIN\_C1 //Reconocido

#DEFINE P4550 PIN\_C2 //Reset Digital

#DEFINE Tipo recbuf[0]

#DEFINE Frec\_4 recbuf[1]

#DEFINE Frec\_3 recbuf[2]

#DEFINE Frec\_2 recbuf[3]

#DEFINE Frec\_1 recbuf[4]

#DEFINE Frec\_0 recbuf[5]

#DEFINE Offset\_3 recbuf[6]

#DEFINE Offset\_2 recbuf[7]

#DEFINE Offset\_1 recbuf[8]

#DEFINE Offset\_0 recbuf[9]

#DEFINE Control recbuf[10]

#byte ANSEL = 0xF7E

#byte ANSELH = 0xF7F

const int8 Lenbuf = 12; // 11 bytes + caracter nulo

int i;

int j;

char recbuf[Lenbuf] = {'0','0','0','0','0','0','0','0','0','0','0'};

char envbuf[Lenbuf] = {'0','0','0','0','0','0','0','0','0','0','0'};

char letra;

void main(void){

ANSEL = 0x00;

ANSELH = 0x00;

Enciende(P4550);

Apaga(LED\_ROJO);

Apaga(LED\_VERDE);

delay\_ms(100);

Enciende(LED\_ROJO);

usb\_init(); //Se inicia el perisferico USB

usb\_task(); //Se configura el perisferico USB

usb\_wait\_for\_enumeration(); //Se espera la respuesta del SO

while (TRUE){

if(usb\_enumerated()){

Apaga(LED\_ROJO);

Enciende(LED\_VERDE);

if (usb\_kbhit(1)){ //Se verifica la existencia de un nuevo paquete en el ENDPOINT

usb\_get\_packet(1, recbuf, Lenbuf); // Recibe Packet

strcpy(envbuf,recbuf);

recbuf[0] = '0';

recbuf[1] = '0';

recbuf[2] = '0';

recbuf[3] = '0';

recbuf[4] = '0';

recbuf[5] = '0';

recbuf[6] = '0';

recbuf[7] = '0';

recbuf[8] = '0';

recbuf[9] = '0';

recbuf[10] = '0';

Apaga(P4550); //Se resetea el PIC18F4550

delay\_ms(5);

Enciende(P4550); //Se inicia el PICF4550

delay\_ms(200); //Se espera al inicio completo del PIC18F4550

for(i=0;i<=10;i++){ //Se inicia el envio de la configuracion de la señal a generar

letra = envbuf[i];

putc(letra);

}

}

}

}

}