ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA

SECCIÓN DE INGENIERÍA INDUSTRIAL

Universidad de La Laguna

MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

Diseño e Implementación de un Instrumento Virtual para la Adquisición y Generación de Señales

Nelson Suárez Martín

Septiembre 2017

Tutores

Oswaldo B. González Hernández & Jonás Philipp Lüke

A todos los familiares y amigos que han estado en todo momento mostrándome su apoyo, así como a los tutores Oswaldo González y Jonás P. Lüke y al profesor Carlos Sanluis, por transmitirme una pequeña parte de sus grandes conocimientos en la materia que a continuación se expone.

¡¡GRACIAS!!

Resumen/Abstract

SGandO, como se le ha denominado coloquialmente al proyecto realizado, comprende una parte hardware y una parte software que, unido a la realización de una PCB, reúne gran parte de los conocimientos adquiridos en una ingeniería industrial, en la rama de electrónica.

Ya sea desde un PC o por medio de la misma placa donde se aloja la FPGA, *SGandO* nos permite generar señales del tipo senoidal, triangular y cuadrada en un amplio rango de frecuencias que van desde los hercios a los megahercios a diferentes magnitudes de voltaje, con la posibilidad de añadir *offset* y modificar el ciclo de trabajo de estas señales.

El osciloscopio nos va a permitir visualizar las señales generadas, bien sean desde el propio generador de señales de *SGandO* o desde fuentes externas, siempre teniendo en cuenta los rangos de entrada permitidos. Entre sus características, cabe destacar la instalación de un amplificador de ganancia programable (PGA) que no distorsiona las señales con bajos niveles de amplitud y la posibilidad de utilizar el nivel del umbral de disparo así como el modo AC para eliminar la componente DC de la señal de entrada.

SGandO, as we have named the completed project, includes hardware and software which, together to the fabrication of a PCB, comprises many topics of acquired knowledge in an Industrial Engineering, in the branch of Electronics.

Either from a PC or through the board where is located the FPGA, SGandO allows us to generate signals of sinusoidal, triangular and square type in a wide range of frequencies ranging from hertz to megahertz with different voltage magnitudes, as well as the possibility of adding offset and modify the duty cycle of these signals.

The oscilloscope will allow us to visualize the generated signals, from either the SGandO signal generator or external sources, always taking into account the allowed input ranges. Its features include the installation of a programmable gain amplifier that does not distort the signals with low amplitude levels and the possibility of using the trigger as well as the AC mode to eliminate the DC component of the input signal.

Lista de Contenidos

Lista de Figuras

Abreviaturas

- **ADC:** Analog-to-Digital Converter.
- **ASCII:** American Standard Code for Information Interchange.
- **CLB:** Configurable Logic Blocks.
- **CMOS:** Complementary Metal–Oxide–Semiconductor.
- **DAC:** Digital-to-Analog Converter.
- **DCM:** Digital Clock Manager.
- **FPGA:** Field Programmable Gate Array.
- **HDL:** Hardware Description Lenguage.
- **IEEE:** Institute of Electrical and Electronics Engineers.
- **I/O:** In/Out.
- **IOB:** Input/Output Blocks.
- **FFT:** Fast Fourier Transform.
- **GPU:** Graphics Processor Unit.
- **LCD:** Liquid Cristal Display.
- **LED:** Light–Emitting Diode.

- **LSB:** Least Significant Bit.
- **LUT:** Look–Up Tables.
- **LVCMOS:** Low Level Complementary Metal–Oxide–Semiconductor.
- **LVTTL:** Low Level Transistor–Transistor Logic.
- **MPSoC:** Multiprocessor System-on-Chip.
- **MSB:** Most Significant Bit.
- **PCB:** Printed Circuit Board.
- **PGA:** Programmable-Gain Amplifier.
- **PNG:** Portable Network Graphics.
- **px:** Píxeles.
- **RAM:** Random Access Memory.
- **RTL:** Register Transfer Level.
- **SMA:** SubMiniature version A.
- **SMD:** Surface Mounted Device.
- **SPI:** Serial Peripheral Interface.
- **SVG:** Scalable Vector Graphics.
- **TTL:** Transistor–Transistor Logic.
- **UART:** Universal Asynchronous Receiver–Transmitter.
- **UCF:** User Constraints File.
- **USB:** Universal Serial Bus.

- **VHDL:** Very High Speed Integrated Circuits – Hardware Description Language.
- **XML:** Extensible Markup Language.

Parte I

Introducción

Capítulo 1

Introducción

1.1. Objetivos

Este proyecto tiene como objetivo la realización de un sistema para la generación de diversos tipos de señales para la alimentación o prueba de otros sistemas electrónicos. Así mismo, se pretende diseñar un software para PC en el que se pueda visualizar las formas de onda de sistemas externos.

Para ello, se ha diseñado mediante código VHDL dos sistemas, generador de señales y osciloscopio, conectados por comunicación serie RS–232 a un software de PC programado en *C++* capaz de manipular diferentes parámetros en los dos sistemas mencionados.

Para la realización de la parte hardware, se configurará un dispositivo semiconductor como es una FPGA disponible en la placa *Spartan3-AN Starter Kit* [1] del desarrollador *Xilinx, Inc.* La aplicación para la configuración de esta placa será por medio del software proporcionado por el propio fabricante, cuyo nombre es (*ISE Design Suite*), mediante código VHDL. La simulación para la depuración de posibles errores se realizará a través de la aplicación *ISim* incluida también en esta herramienta para, finalmente, hacer uso de otra de las aplicaciones incluidas en el software del desarrollador, el *ISE Impact*, para cargar la configuración en la FPGA de la placa.

El diseño del software se llevará a cabo mediante la aplicación *Qt Designer* [2]. La interfaz gráfica, programada en código *C++*, contendrá los mandos para el manejo tanto

del generador de señales como del osciloscopio.

Debido a las limitaciones técnicas de los componentes electrónicos incluidos en la *Spartan3-AN Starter Kit* tales como el convertidor digital – analógico (DAC) o el convertidor analógico – digital (ADC), se ha decidido llevar a cabo la fabricación de una placa que contenga una versión mejorada de estos componentes, así como sistemas de amplificación de señales, como puede ser un amplificador diferencial para la salida del DAC o un amplificador programable o PGA para la entrada previa al ADC.

1.2. Estructura de la memoria

La memoria del proyecto que en este documento se describe consta de un total de nueve capítulos más los diferentes anexos que al final del mismo se adjuntan.

El primer capítulo consta de una breve introducción dividida en tres sub-apartados: objetivos del proyecto que se realiza, estructura de la memoria que es el apartado donde nos encontramos y, por último, la planificación del proyecto inicial concluyendo con las causas en el retraso del mismo.

El capítulo dos integra todo lo referente a recursos tanto hardware como software empleados en el proyecto. El primer punto de este capítulo es una breve introducción al VHDL como código de descripción hardware, continuando por la descripción de la placa utilizada, la *SPARTAN–3AN Starter Kit* [1], incluyendo aquí los principales componentes y actuadores utilizados. Aún dentro de este capítulo, se describe en el siguiente punto las aplicaciones software utilizadas:

- *ISE Design Suite 14.7* para la descripción hardware en VHDL.
- *ISim 14.7* para la simulación de los diferentes módulos en VHDL.
- *ISE iMPACT* para la configuración de la FPGA.
- *Qt* para la programación de la aplicación software para PC.
- *Inkscape* para el diseño de botones e iconos de la aplicación software en Qt.

- *KiCad* para el diseño electrónico del circuito y PCB externe a la placa de *Xilinx*.
- *Realterm* para la comprobación de la comunicación entre el PC y la FPGA.

El tercer capítulo versa de la descripción del proyecto realizado como Trabajo Fin de Máster o TFM. Se comienza por la descripción de la parte hardware, que es todo lo relacionado con el lenguaje VHDL y las aplicaciones de *Xilinx, Inc.*, explicando cada uno de los módulos diseñados y su funcionamiento. En el segundo punto dentro de este mismo capítulo, se explica también el funcionamiento y el cómo se diseño la aplicación software con *Qt Design*. Por último, se describe la placa de circuito impreso o PCB que se ha realizado y que fue necesaria para llevar a cabo el proyecto actual.

El capítulo cuatro se destina a la experimentación realizada una vez se ha finalizado el proyecto, comparando lo que se ha hecho con aparatos de laboratorio que realizan las mismas acciones. En este capítulo, se incluyen también las conclusiones con el resumen y las propuestas de mejora al proyecto.

Para finalizar la memoria de este proyecto, se incluyen las referencias que se han citado durante la redacción del mismo, así como el apartado destinado al presupuesto. Además, se han añadido en los anexos las fichas de datos de todos los componentes utilizados en la realización de la PCB.

1.3. Planificación

La realización de la planificación de este proyecto se ha llevado a cabo con el software libre *GanttProject* [3], una aplicación de escritorio multiplataforma para la programación y gestión de proyectos. Dicha planificación es la mostrada en la figura 1.1.

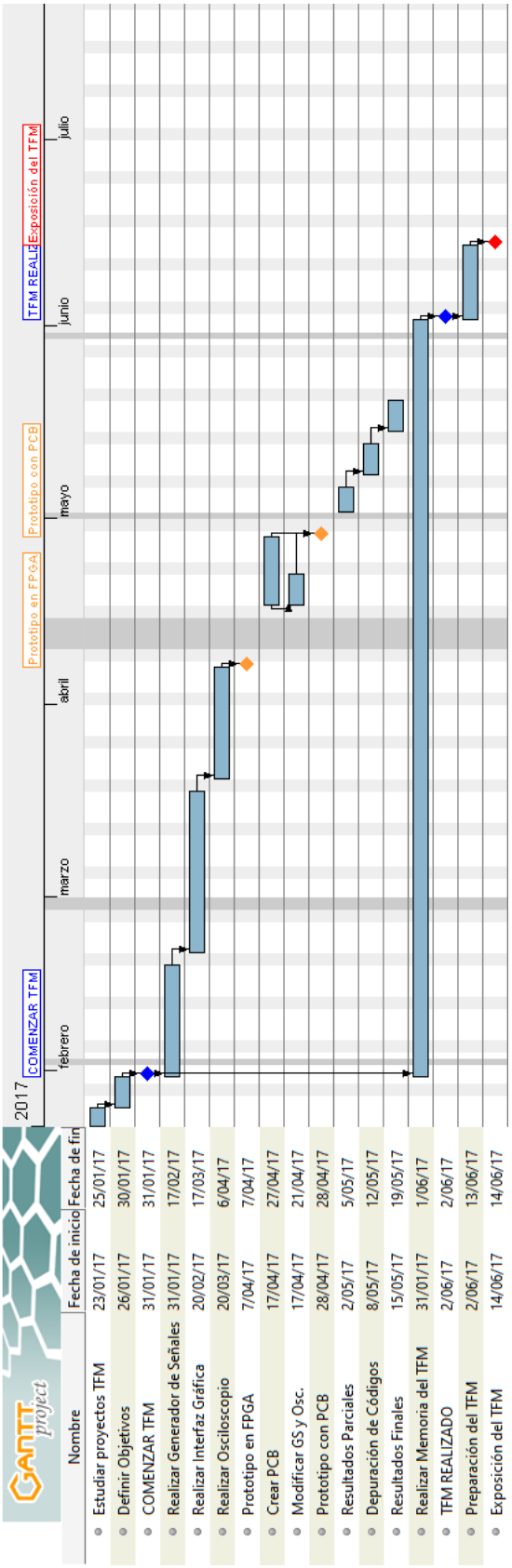


Figura 1.1: Cronograma del proyecto SGandO realizado en *GanttProject*. *Elaboración propia.*

La planificación inicial del *Trabajo Fin de Máster* (TFM) empieza el *23/01/2017* con el estudio de los posibles proyectos a realizar. Una vez se supo la idea y se concretaron objetivos, el *31/01/2017* se comienza. Los hitos que se marcaron fueron:

- **Prototipo en FPGA:** Para el cumplimiento de este primer hito, debieron realizarse dentro del tiempo estimado los procesos que darían lugar al generador de señales (dos semanas). Una vez comprobado el funcionamiento del mismo, utilizando únicamente la placa *Spartan3-AN Starter Kit* [1], se prosiguió con la realización de la interfaz gráfica, calculada para hacerse en un tiempo estimado de *18 días*. En último lugar, se calcularon dos semanas para realizar el osciloscopio. En total, la consecución del hito *Prototipo en FPGA* se estimó en mes y medio, un número que se vio incrementado en la fase de diseño de la interfaz gráfica debido al alto número de gráficos realizados en *Inkscape* [4] para dar vida a la misma.

- **Prototipo con PCB:** Para este segundo hito se estimó una duración total de una semana y media, segregados en *Crear PCB* y *Modificar Generador de Señales y Osciloscopio*, habiendo un importante retraso en el primero de ellos debido, no tan solo al pedido de componentes, sino a las complicaciones a la hora de poner en marcha la placa con componentes SMD.

- **TFM REALIZADO:** El retraso del hito anterior ocasionó que la duración estimada de tres semanas para terminar el proyecto (2 de junio), incluyendo la finalización de la memoria del mismo, se extendiese hasta los dos meses. Este último hito se segregó en *Resultados Parciales*, *Depuración de Códigos*, donde se incluyen los ficheros VHDL y *C++*, y *Resultados Finales*.

Parte II

Recursos Hardware/Software Utilizados

Capítulo 2

VHDL

VHDL es un lenguaje de descripción hardware definido por el IEEE usado por ingenieros para la descripción tanto estructural como funcional de circuitos digitales.

2.1. Diseño Top-Down

Este método de diseño es el que se trata de emplear en este proyecto usando el lenguaje VHDL y es el implementado por el software utilizado (*ISE Design Suite 14.7*). Su principal característica es que, partiendo de una descripción bastante abstracta, se puede ir bajando de niveles a medida que vamos aumentando el detalle del diseño del proyecto hasta llegar al nivel más bajo que necesitemos, por ejemplo, la descripción a nivel de puertas lógicas.

Las principales ventajas de la utilización de este diseño son:

- **Incrementa la productividad del diseño**, es decir, podemos especificar mediante código el diseño de un circuito y el software destinado a la implementación del mismo generaría el nivel de puertas lógicas automáticamente.

- **Incrementa la reutilización del diseño**, ya que en el proceso se utilizan estándares que permiten reutilizar los códigos aun cambiando de herramienta para la implementación, siendo posible crear diseños nuevos con códigos ya existentes.

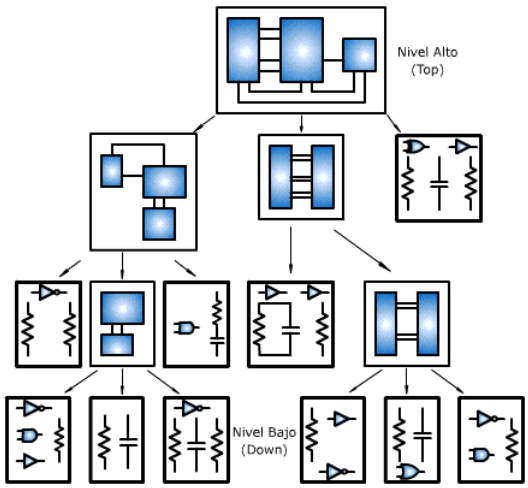


Figura 2.1: Método de diseño *Top–Down*.

- **Rápida detección de errores** debido a la estructura por bloques del diseño.

En la figura 2.1, se presenta este diseño para el proyecto llevado a cabo.

2.2. Estilos de descripción en VHDL

En el lenguaje de descripción hardware VHDL son posibles varios estilos de descripción dependiendo del nivel de abstracción.

2.2.1. Descripción flujo de datos

También conocida como descripción de transferencia entre registros (RTL, *Register Transfer Level*), es la descripción más cerca al hardware real. En este tipo de descripción, la ejecución de las sentencias es concurrente, es decir, dichas sentencias actúan como conexiones o leyes que se ejecutarán al mismo tiempo en todo el proyecto, entendiendo como conexión a las interconexiones entre objetos abstractos. Las instrucciones serán siempre de asignación, siendo los datos los que gobiernen el flujo de ejecución de las instrucciones. La asignación condicional *WHEN–ELSE* o la referencia a entidades o componentes en la arquitectura de un código son claros ejemplos para la definición de instrucciones en RTL.

2.2.2. Descripción algorítmica

VHDL también permite, al igual que la mayoría de los lenguajes de programación, una programación serie o algorítmica para la descripción de códigos que nos permita utilizar sentencias condicionales tales como *IF–THEN–ELSE*, de selección como *CASE* o bucles *LOOP*, *FOR* o *WHILE*. Esta programación en VHDL se hará siempre dentro de bloques indicados por la palabra clave *PROCESS* y es posible tener varios de estos procesos dentro de una misma arquitectura comportándose de manera concurrente. Cada proceso se podrá ejecutar de manera independiente cuando se produzca un evento en las señales de la lista de sensibilidad o en las sentencias *WAIT*. La estructura del bloque de ejecución serie *PROCESS* es la siguiente:

```
1 [id_process:] PROCESS [(lista_sensible)] [IS]
2 --Declaración de variables
3 BEGIN
4 --Instrucciones serie
5 END PROCESS [id_process];
```

El código entre corchetes '[...]' es opcional, teniendo en cuenta que si no definimos una lista de sensibilidad es necesaria la utilización de sentencias *WAIT* dentro del proceso porque si no, no se llegaría a ejecutar nunca.

En los bloques *PROCESS* tendremos en cuenta la diferencia existente entre señal y variable, la cual radica en que la **variable** cambia su valor en el momento de asignación mientras que la **señal** actualizará su valor una vez se haya ejecutado todo el proceso, por lo que **cambios intermedios no se verán reflejados, únicamente la última asignación**.

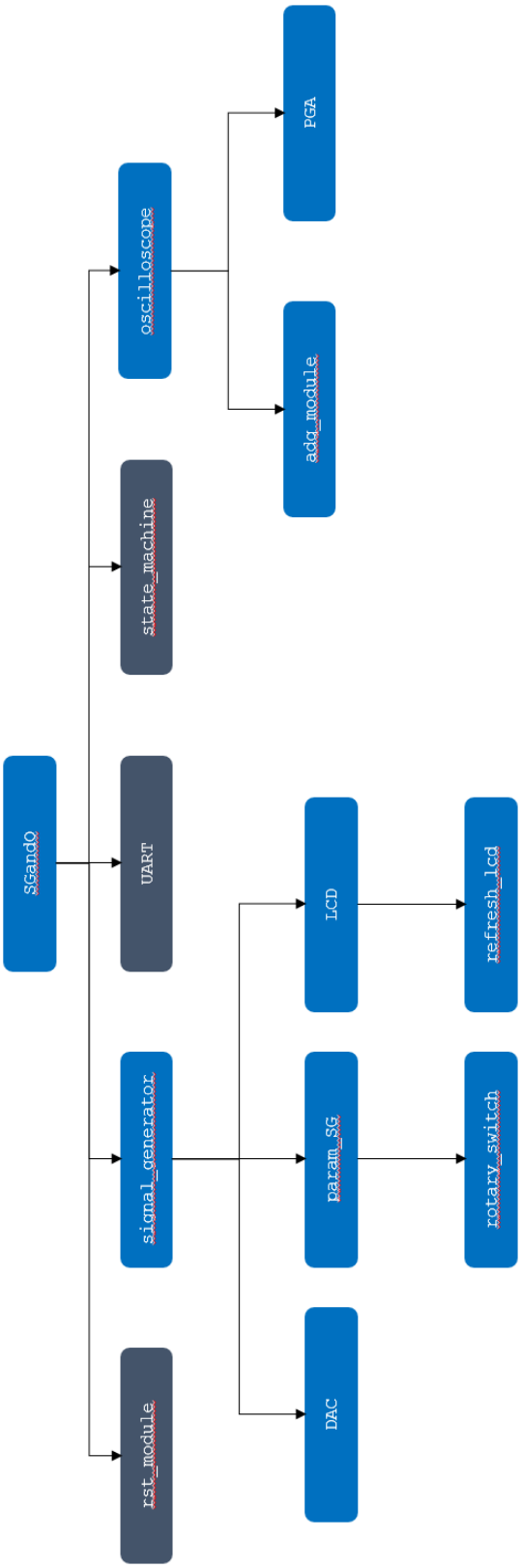


Figura 2.2: Diseño *Top–Down* del proyecto. *Elaboración propia.*

2.2.3. Descripción estructural

Siendo las descripciones anteriormente citadas las más usadas en lenguaje de descripción hardware, hay veces que se hace necesario una descripción estructural para el interconexionado de componentes, sobre todo cuando se trata de un proyecto grande con muchos bloques. Es un estilo más cercano al *Netlist*, que consiste en dar una lista de componentes, sus interconexiones y las entradas y salidas. En VHDL y para este proyecto hemos usado los siguientes mecanismos donde podemos ver el uso de la descripción estructural:

- **Componentes:** Con ellos haremos referencia a una entidad desde otra. La palabra clave para su definición es *COMPONENT* y su estructura será como sigue:

```
1 COMPONENT nombre_componente [IS]
2 [GENERIC(lista_de_parámetros);]
3 [PORT(lista_de_puertos);]
4 END COMPONENT [nombre_componente];
```

  El nombre del componente será el mismo que el de la entidad al cual hace referencia, al igual que sus parámetros genéricos y puertos.

- **Repetición de estructuras:** En muchas ocasiones se nos presentará el caso de tener que repetir un determinado componente. Para ello haremos uso de la sentencia *GENERATE*, la cual presenta la siguiente estructura:

```
1 Id_generate: FOR/IF parámetro_repetitivo/condición GENERATE
2 [declaraciones
3 BEGIN]
4 [sentencias]
5 END GENERATE [id_generate];
```

  En este caso, la etiqueta *id_generate* es **obligatoria**.

- **Conexiones:** Por último, existe la posibilidad de conectar varios componentes haciendo referencia a los mismos mediante *GENERIC MAP* y *PORT MAP* (figura 2.3). Un ejemplo de esto último es:

```
Id_1: nombre_componente GENERIC MAP (parámetros)
PORT MAP (entrada, a, b);
Id_2: nombre_componente GENERIC MAP (parámetros)
PORT MAP (a, b, salida);
```

Figura 2.3: Conexión de componentes. *Elaboración propia.*

Capítulo 3

Spartan–3AN Starter Kit

Para llevar a cabo el diseño del proyecto se ha hecho uso de una placa de desarrollo basada en una FPGA de la familia *Spartan–3AN* del fabricante *Xilinx, Inc.*

La tarjeta de desarrollo consta de múltiples componentes, entre los cuales caben destacar los siguientes:

- FPGA Spartan-3AN XC3S700AN - FGG484.
- Puerto Serie RS-232.
- Pantalla LCD 2x16.
- Reloj de 50 MHz.
- Selectores.
- Pulsadores.
- Indicadores LED.

La tarjeta se programa a través de un cable USB Tipo A/Tipo B por el puerto destinado en la placa para ello.

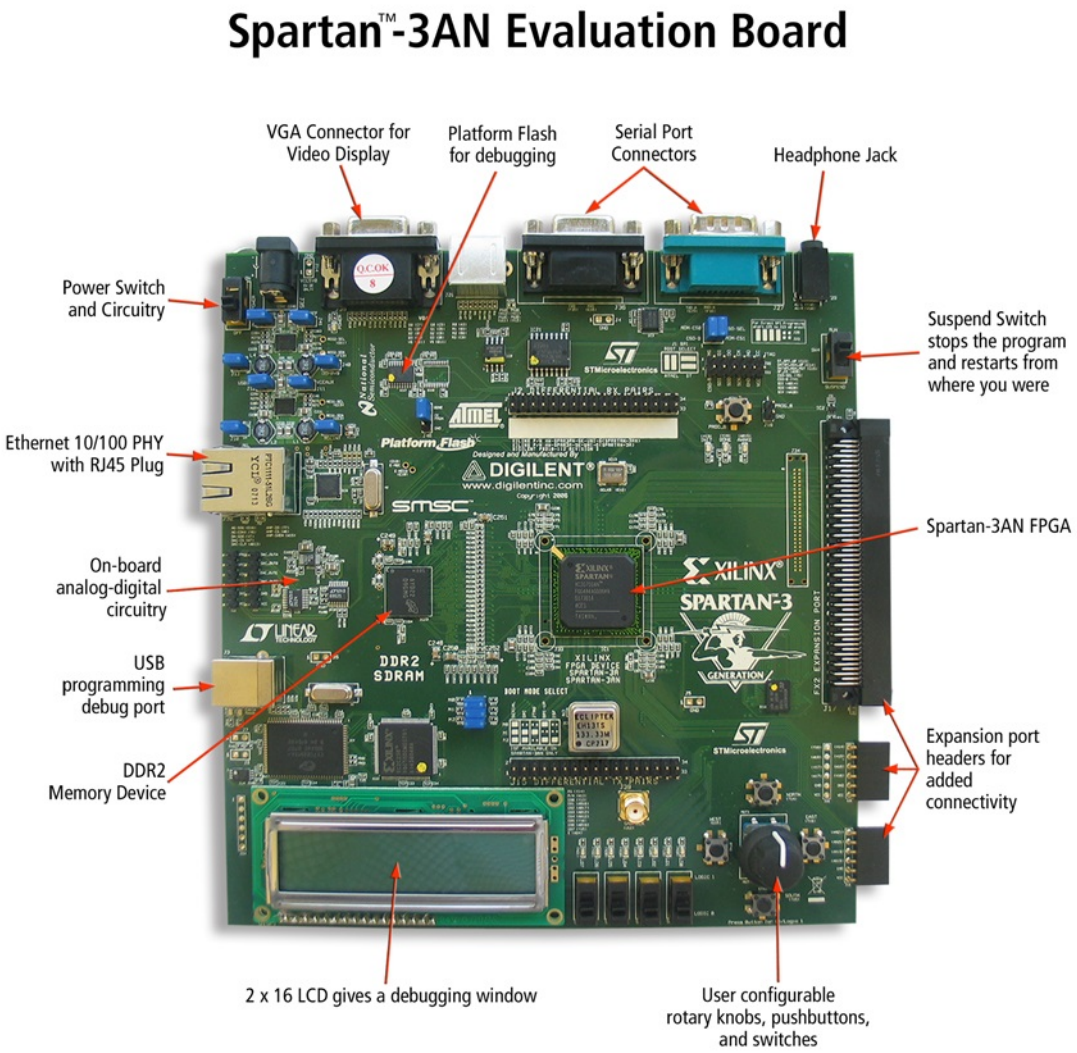


Figura 3.1: *Spartan–3AN Starter Kit Board* [1].

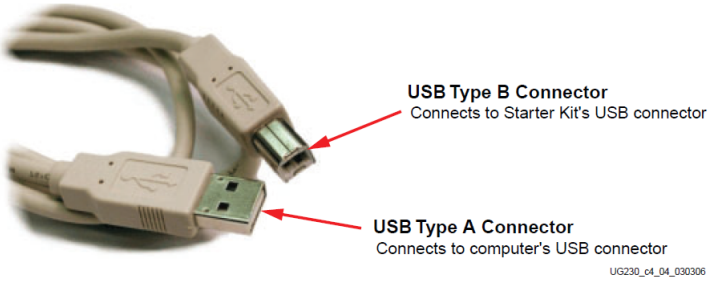


Figura 3.2: Cable de programación de placa *Spartan–3AN Starter Kit* [1].

3.1. FPGA Spartan–3AN

En la figura 3.3 se muestra un esquema básico de la FPGA utilizada, la cual presenta la arquitectura de la figura 3.4.

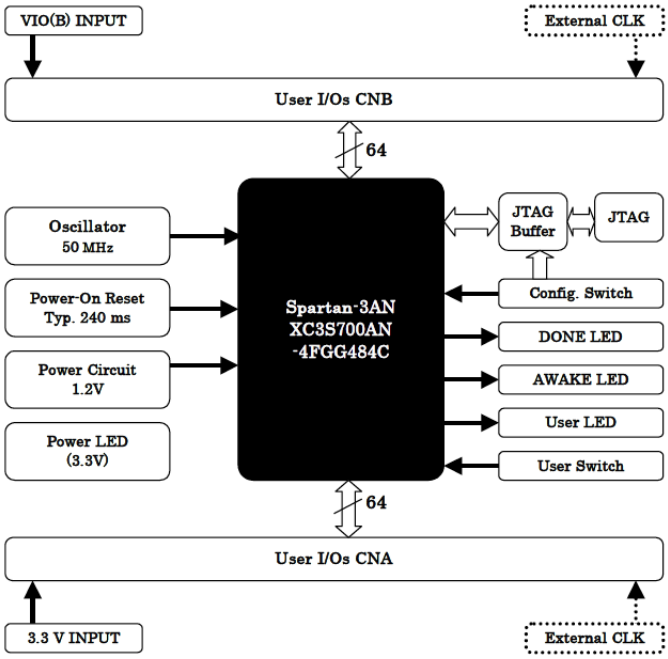


Figura 3.3: Esquema básico de la FPGA *Spartan–3AN* [5].

En esta arquitectura existen cinco elementos fundamentales que citaremos a continuación:

- **Configurable Logic Blocks (CLB):** Contienen las *Look–Up Tables* (LUT) o tablas de consulta donde se implementa la lógica programable, además de ser los bloques que incluyen elementos como flip-flops o registros. Contienen una amplia variedad de funciones lógicas además de almacenamiento de datos.
- **Input/Output Blocks (IOB):** Es donde se controla el flujo de datos entre los pines I/O y la lógica interna.
- **Block RAM:** Proporciona el almacenamiento de los datos en bloques de dos puertos de 18 kbits cada uno.

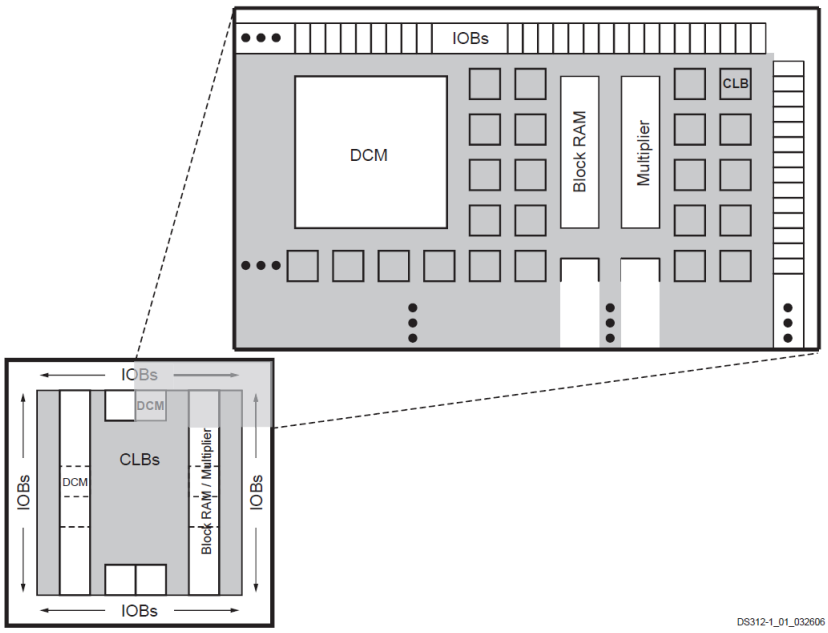


Figura 3.4: Arquitectura de la FPGA *Spartan–3AN* [6].

- **Multiplier Blocks:** Calcula el producto de dos números binarios de 18 bits cada uno existentes a la entrada del bloque. Cada multiplicador está asociado a un bloque RAM.

- **Digital Clock Manager (DCM) Blocks:** Es un bloque dedicado a la auto–calibración así como divisores, multiplicadores o cambio de fase en las señales de entrada de reloj.

En la figura 3.5 podemos ver el *footprint* de la FPGA con la distribución de sus 484 conexiones (tres de ellas desconectadas en nuestro modelo XC3S700AN marcadas con un rombo) y de donde surgen las asignaciones de pines que tendremos que establecer en el archivo *User Constraints File* (UCF).

3.2. Jumpers de configuración

La placa dispone de *jumpers* para la alimentación de algunos de los pines de la FPGA así como para la configuración de memoria a la hora de programar nuestro código. La

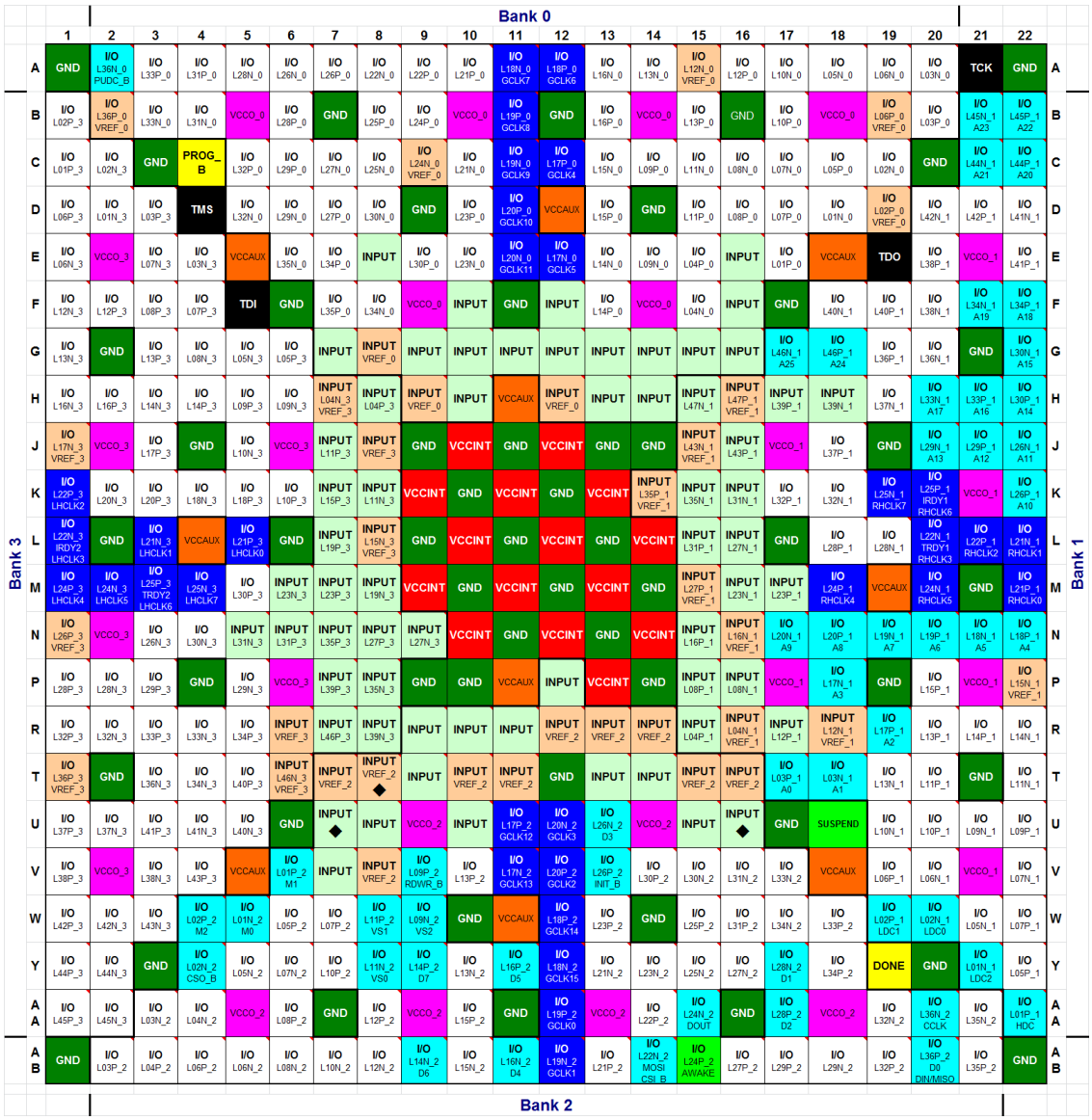


Figura 3.5: Footprint de la FPGA *Spartan–3AN* [6].

disposición de estos *jumpers* es la dispuesta por defecto por el fabricante (figura 3.6).

Los *jumpers* destinados a la alimentación estarán siempre conectados mientras que la distribución de los mismos destinados a la configuración de memoria podemos cambiarla según las características del proyecto que llevemos a cabo. Las diferentes configuraciones podemos verlas en la figura 3.7 que mostramos a continuación, estando seleccionada la utilizada en este proyecto y que a continuación se explica.

Para el proyecto que se ha llevado a cabo no se vio la necesidad de emplear memoria

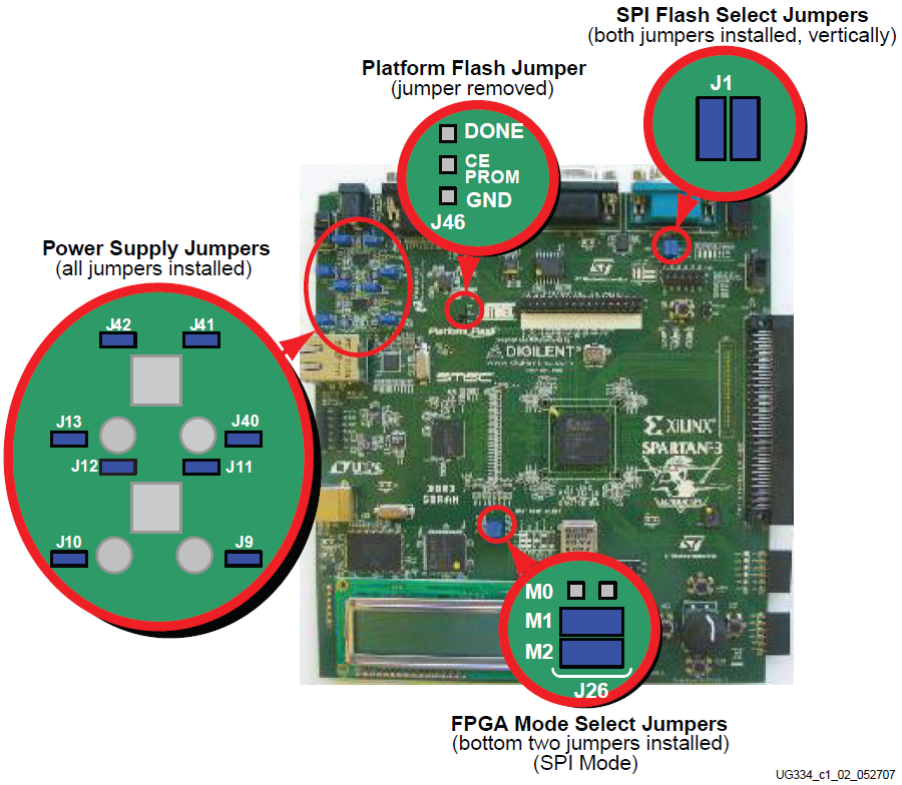


Figura 3.6: Configuración de los *jumpers* por defecto [1].

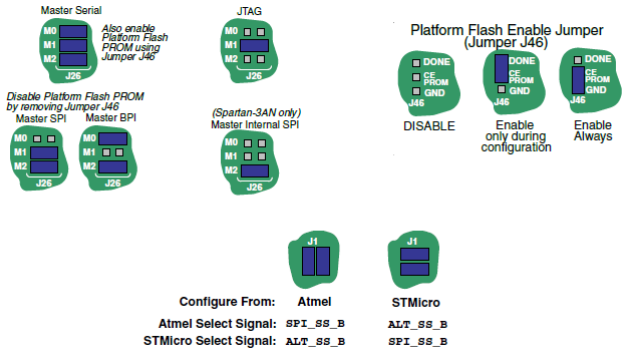


Figura 3.7: Disposición de *jumpers* para la configuración de la placa [7].

flash debido a que el código a programar es relativamente corto y, por consiguiente, la fase de programación instantánea para la comprobación del mismo es rápida. Aun así, a continuación procederemos a explicar el tipo de memoria seleccionada y sus principales características.

- **Jumper J26:** Con este *jumper* indicamos la manera de grabar el código perma-

nentemente en la placa por medio de la memoria flash, siendo la elegida la *Master SPI*.

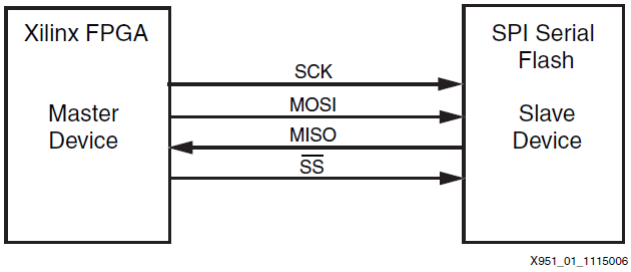


Figura 3.8: Diagrama básico de la configuración de memoria *flash* SPI [8].

SCK (*Clock Signal*): El dispositivo maestro (*Xilinx FPGA*) es el encargado de marcar las pautas en la sincronización con el dispositivo esclavo (*SPI Serial Flash*).

MOSI (*Master Out Slave In*): Cuando SS (*Slave Select*) está a nivel bajo o *low* el dispositivo maestro escribe en el esclavo.

MISO (*Master In Slave Out*): En el mismo ciclo de reloj SCK en el que escribe el maestro, el esclavo devuelve el mismo dato por MISO.

SS (*Slave Select*): Señal para el control del flujo de datos entre MOSI y MISO. Cuando esta está en nivel bajo o *low*, el dispositivo esclavo está 'habilitado' para la transmisión de datos, mientras que cuando su valor está en nivel alto o *high* el dispositivo esclavo estará 'deshabilitado'.

Más información acerca de la configuración *Master SPI* en el documento [8].

- **Jumper J1:** Este *jumper* indicará cuál de las memorias de la placa es el esclavo principal para la configuración SPI y cuál es el esclavo secundario. En nuestro caso, el principal es la del fabricante *Atmel* [9] y la secundaria *STMicro* [10]. En la siguiente figura, se muestra la localización de las memorias en la placa.

- **Jumper J46:** Si estamos usando el modo de memoria *Flash Master SPI* es imprescindible desconectar este *jumper* según instrucciones en el documento citado en las referencias del documento [1] en la página 94.

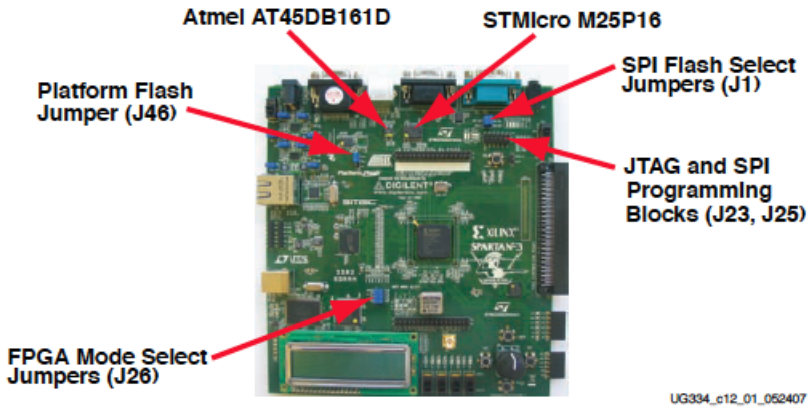


Figura 3.9: Localización de las memorias en la placa *Spartan-3AN Starter Kit* [1].

3.3. Conector Hirose 100-P

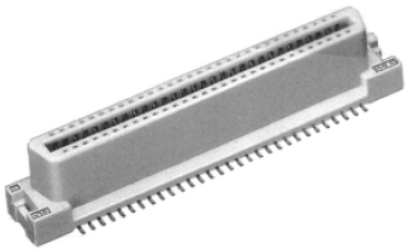


Figura 3.10: Puerto de expansión FX2-100p [1].

La realización del proyecto actual lleva consigo la utilización del puerto de expansión de 100 pines de la placa de *Xilinx*.

Los 100 pines del conector se distribuyen en:

- 45 pines a GND.
- 40 pines de I/O.
- 10 pines no configurables.
- 3 pines con alimentación 5V.
- 2 pines con alimentación 3V3.

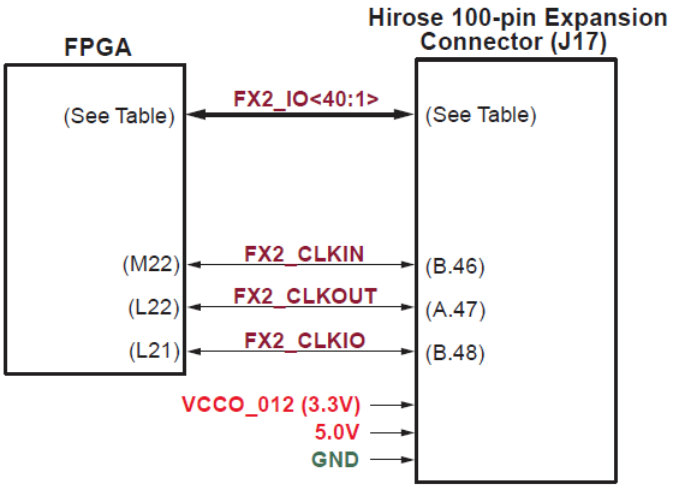


Figura 3.11: Puerto de expansión FX2-100p [1].

Más información acerca de los pines de este conector se puede encontrar en el manual de la placa de *Xilinx* [1]. Así mismo, a continuación se detalla la asignación de los 40 pines de I/O en el fichero de restricciones.

```
# ==== FX2 Connector (FX2) ====
NET "FX2_IO<1>" LOC = "A13" | IOSTANDARD = LVCMOS33 | SLEW =
   FAST | DRIVE = 8;
NET "FX2_IO<2>" LOC = "B13" | IOSTANDARD = LVCMOS33 | SLEW =
   FAST | DRIVE = 8;
NET "FX2_IO<3>" LOC = "A14" | IOSTANDARD = LVCMOS33 | SLEW =
   FAST | DRIVE = 8;
NET "FX2_IO<4>" LOC = "B15" | IOSTANDARD = LVCMOS33 | SLEW =
   FAST | DRIVE = 8;
NET "FX2_IO<5>" LOC = "A15" | IOSTANDARD = LVCMOS33 | SLEW =
   FAST | DRIVE = 8;
NET "FX2_IO<6>" LOC = "A16" | IOSTANDARD = LVCMOS33 | SLEW =
   FAST | DRIVE = 8;
NET "FX2_IO<7>" LOC = "A17" | IOSTANDARD = LVCMOS33 | SLEW =
   FAST | DRIVE = 8;
NET "FX2_IO<8>" LOC = "B17" | IOSTANDARD = LVCMOS33 | SLEW =
```

```
    FAST | DRIVE = 8;
10  NET "FX2_IO<9>" LOC = "A18" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
11  NET "FX2_IO<10>" LOC = "C18" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
12  NET "FX2_IO<11>" LOC = "A19" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
13  NET "FX2_IO<12>" LOC = "B19" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
14  NET "FX2_IO<13>" LOC = "A20" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
15  NET "FX2_IO<14>" LOC = "B20" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
16  NET "FX2_IO<15>" LOC = "C19" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
17  NET "FX2_IO<16>" LOC = "D19" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
18  NET "FX2_IO<17>" LOC = "D18" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
19  NET "FX2_IO<18>" LOC = "E17" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
20  NET "FX2_IO<19>" LOC = "D20" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
21  NET "FX2_IO<20>" LOC = "D21" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
22  NET "FX2_IO<21>" LOC = "D22" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
23  NET "FX2_IO<22>" LOC = "E22" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
24  NET "FX2_IO<23>" LOC = "F18" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
```

```
25 NET "FX2_IO<24>" LOC = "F19" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
26 NET "FX2_IO<25>" LOC = "F20" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
27 NET "FX2_IO<26>" LOC = "E20" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
28 NET "FX2_IO<27>" LOC = "G20" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
29 NET "FX2_IO<28>" LOC = "G19" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
30 NET "FX2_IO<29>" LOC = "H19" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
31 NET "FX2_IO<30>" LOC = "J18" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
32 NET "FX2_IO<31>" LOC = "K18" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
33 NET "FX2_IO<32>" LOC = "K17" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
34 NET "FX2_IO<33>" LOC = "K19" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
35 NET "FX2_IO<34>" LOC = "K20" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
36 NET "FX2_IO<35>" LOC = "L19" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
37 NET "FX2_IO<36>" LOC = "L18" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
38 NET "FX2_IO<37>" LOC = "M20" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
39 NET "FX2_IO<38>" LOC = "M18" | IOSTANDARD = LVCMOS33 | SLEW =
     FAST | DRIVE = 8;
40 NET "FX2_IO<39>" LOC = "L20" | IOSTANDARD = LVCMOS33 | SLEW =
```

```
    FAST | DRIVE = 8;
41  NET "FX2_IO<40>" LOC = "P20" | IOSTANDARD = LVCMOS33 | SLEW =
    FAST | DRIVE = 8;
```

3.4. Puerto serie RS–232

La *Spartan-3AN FPGA Starter Kit Board* dispone de dos puertos serie *RS–232*, uno hembra (DB9 DCE), que es el que vamos a utilizar y se suele emplear para la comunicación con el PC, y otro macho (DB9 DTE) que se usará en caso de que queramos conectar otros periféricos con salida *RS–232* como pueden ser módems o impresoras, pero que no es nuestro caso.

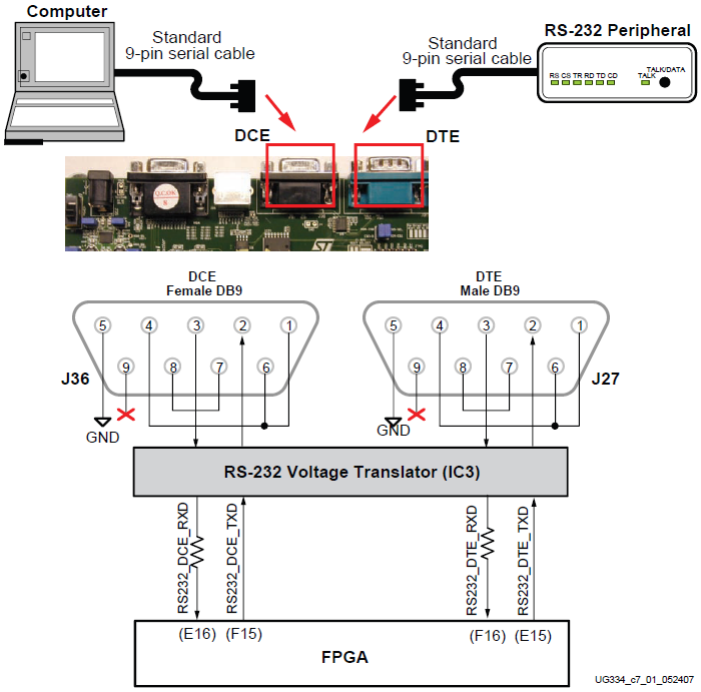


Figura 3.12: Puerto serie RS–232 [1].

La FPGA suministra sus datos a niveles de LVTTL o *Low Level TTL*, o LVCMOS o *Low Level CMOS*, y el *RS–232 Voltage Translator* convierte este valor en niveles adecuados de voltaje para el *RS–232*. Cuando el PC quiere escribir en la FPGA, el

proceso de conversión es el inverso al explicado anteriormente.

La asignación de pines en el fichero UCF es:

```
NET "RS232_DCE_RXD" LOC = E16 | IOSTANDARD = LVCMOS33;
NET "RS232_DCE_TXD" LOC = F15 | IOSTANDARD = LVCMOS33 | DRIVE =
   8 | SLEW = SLOW;
```

3.5. Display LCD

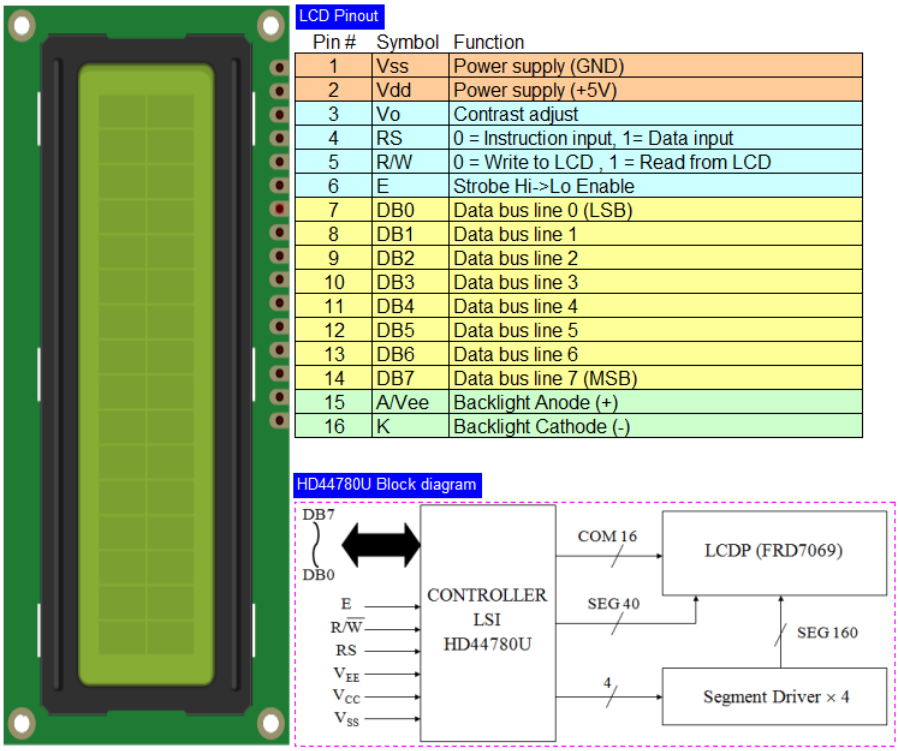


Figura 3.13: Display LCD de la placa de *Xilinx* [11].

El display LCD de la que dispone la placa está controlada mediante la FPGA (figura 3.14), siendo posible suministrar datos en grupos de 8 bits, que es nuestro caso, o bien sólo mediante los 4 bits más significativos teniendo en cuenta que los 4 bits restantes, los menos significativos, tendrán que estar a nivel lógico alto. Para mejorar las características de la pantalla se puede hacer uso de un procesador *PicoBlaze* pero, para las necesidades

de nuestro proyecto, hemos optado por el control mediante la FPGA.

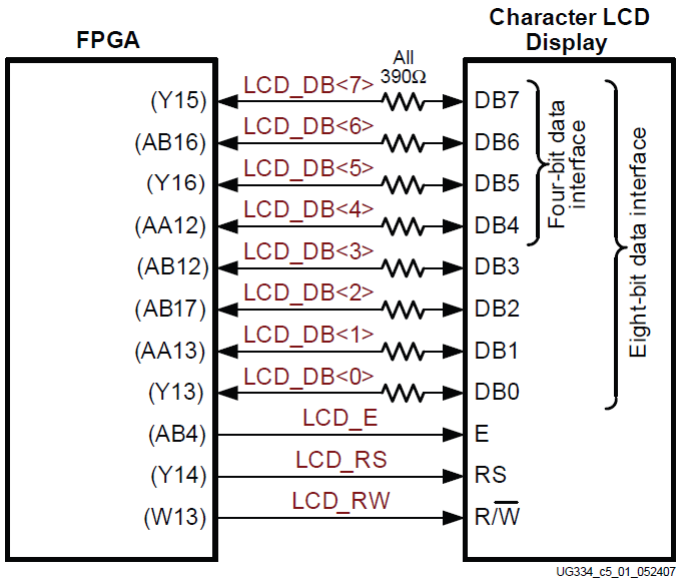


Figura 3.14: Interfaz de la pantalla LCD [1].

Además de las líneas destinadas a los datos ($LCD\_DB < X >$) la pantalla es controlada mediante tres líneas de control ($LCD\_E/RS/RW$).

- **Líneas de datos** $LCD\_DB < X >$**:** La pantalla LCD admite un total de 80 caracteres ASCII [12] de los cuales 32 serán los que se puedan visualizar al mismo tiempo (figura 3.15).

  Línea 1: $0x00 \rightarrow 0x0F$

  Línea 2: $0x40 \rightarrow 0x4F$

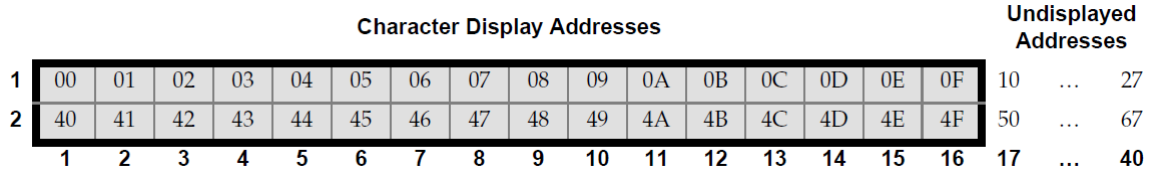


Figura 3.15: Caracteres de la pantalla LCD de la placa [1].

- **Línea de habilitación de lectura y escritura** LCD_E**:**

  0: *Read/Write* deshabilitado.

1: *Read/Write* habilitado.

- **Línea de selección de registro** LCD_RS**:**

  0: Instrucciones de registro durante las operaciones de escritura. La memoria *flash* permanecerá ocupada durante las operaciones de lectura.

  1: Datos para las instrucciones de lectura y escritura.

- **Línea de control de lectura y escritura** LCD_RW**:**

  0: Escribe; la pantalla admite datos.

  1: Lee; la pantalla muestra los datos.

La asignación de pines de la pantalla LCD necesaria para el fichero de restricciones UCF es la siguiente:

```
NET "LCD_E" LOC = AB4 | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW
    = SLOW;
NET "LCD_RS" LOC = Y14 | IOSTANDARD = LVCMOS33 | DRIVE = 8 |
   SLEW = SLOW;
NET "LCD_RW" LOC = W13 | IOSTANDARD = LVCMOS33 | DRIVE = 8 |
   SLEW = SLOW;
NET "LCD_DB<7>" LOC = Y15 | IOSTANDARD = LVCMOS33 | DRIVE = 8 |
   SLEW = SLOW;
NET "LCD_DB<6>" LOC = AB16 | IOSTANDARD = LVCMOS33 | DRIVE = 8 |
    SLEW = SLOW;
NET "LCD_DB<5>" LOC = Y16 | IOSTANDARD = LVCMOS33 | DRIVE = 8 |
   SLEW = SLOW;
NET "LCD_DB<4>" LOC = AA12 | IOSTANDARD = LVCMOS33 | DRIVE = 8 |
    SLEW = SLOW;
NET "LCD_DB<3>" LOC = AB12 | IOSTANDARD = LVCMOS33 | DRIVE = 8 |
    SLEW = SLOW;
NET "LCD_DB<2>" LOC = AB17 | IOSTANDARD = LVCMOS33 | DRIVE = 8 |
    SLEW = SLOW;
```

```
10 NET "LCD_DB<1>" LOC = AB18 | IOSTANDARD = LVCMOS33 | DRIVE = 8 |
       SLEW = SLOW;
11 NET "LCD_DB<0>" LOC = Y13 | IOSTANDARD = LVCMOS33 | DRIVE = 8 |
      SLEW = SLOW;
```

Para más información acerca del funcionamiento de la pantalla LCD descrita anteriormente, ver documento en la cita [13].

3.6. Oscilador 50 MHz

La señal de reloj base utilizada en el proyecto es suministrada por el oscilador de cristal disponible en la placa, CLK_50MHZ (figura 3.16), de $50MHz \pm 2,5kHz$ de frecuencia y asignada al pin E12 de la FPGA. Por lo tanto, no son usadas la señal de reloj auxiliar (CLK_AUX) ni ninguna señal de reloj externa introducida por la entrada CLK_SMA.

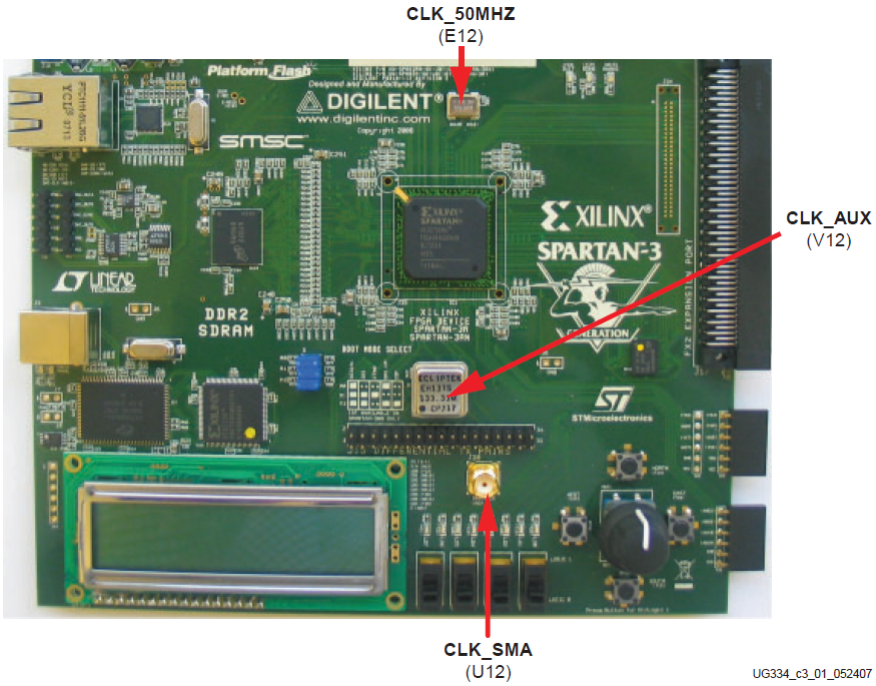


Figura 3.16: Osciladores instalados en la placa [1].

Su asignación en el fichero UCF es:

```
NET "CLK_50MHZ" LOC = E12 | IOSTANDARD = LVCMOS33;
```

3.7. Selectores

- **Selectores configurables:** En la placa disponemos de diferentes selectores, bien sea para la manipulación de nuestro proyecto o bien, para el funcionamiento de la placa. Los selectores que podemos utilizar son los mostrados a continuación, donde se indica también el pin de la FPGA al que están conectados.

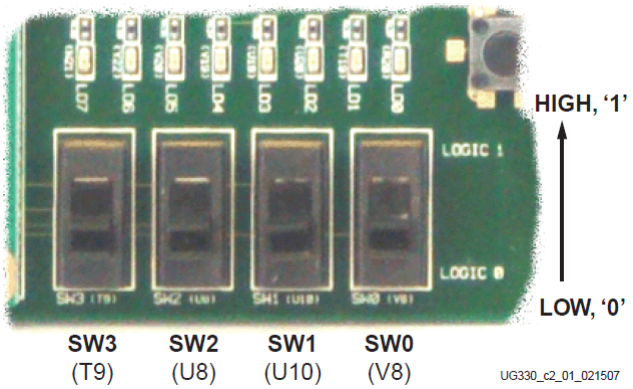


Figura 3.17: Selectores configurables instalados en la placa [1].

La especificación de una resistencia de *pull–up* es opcional en el UCF, aunque, si dispusiéramos del mismo, evitaríamos posibles errores en la transición de los valores del selector. Su definición, cuando queremos especificar dichas resistencias, es la que sigue:

```
NET "SW<0>" LOC = V8 | IOSTANDARD = LVCMOS33;
NET "SW<1>" LOC = U10| IOSTANDARD = LVCMOS33;
NET "SW<2>" LOC = U8 | IOSTANDARD = LVCMOS33;
NET "SW<3>" LOC = T9 | IOSTANDARD = LVCMOS33;
```

- **Selector de modo de suspensión:** Además de los selectores anteriormente cita-

dos, disponemos de otros dos: arranque o suspensión del programa que se encuentra en la FPGA (figura 3.18) y encendido y apagado de la placa.

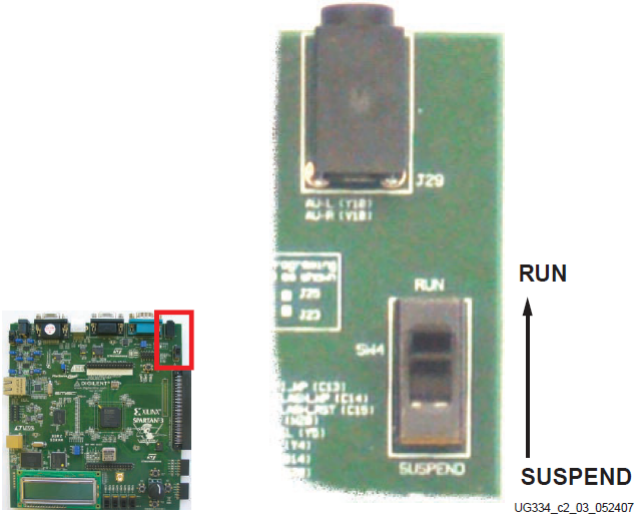


Figura 3.18: Selector de arranque o suspensión del programa [1].

Para la utilización del modo de suspensión antes deberemos habilitarlo en el UCF como sigue a continuación:

```
1 CONFIG ENABLE_SUSPEND = "FILTERED";
```

En caso de tener habilitada la suspensión del programa en funcionamiento y el selector en la posición *'SUSPEND'*, el led marcado en la placa como *'AWAKE'* (figura 3.22) se encenderá como muestra de que está activada. El programa en funcionamiento entrará en modo suspensión y cuando lo reanudemos, selector puesto en la posición *'RUN'*, el programa volverá al estado anterior a cuando hayamos hecho la suspensión. Más información acerca del modo en suspensión en el enlace [14] del anexo *Referencias*.

3.8. Pulsadores

Como ocurre con los selectores, en la placa encontraremos pulsadores que podremos configurar para el proyecto o utilizarlos para el funcionamiento de la propia placa.

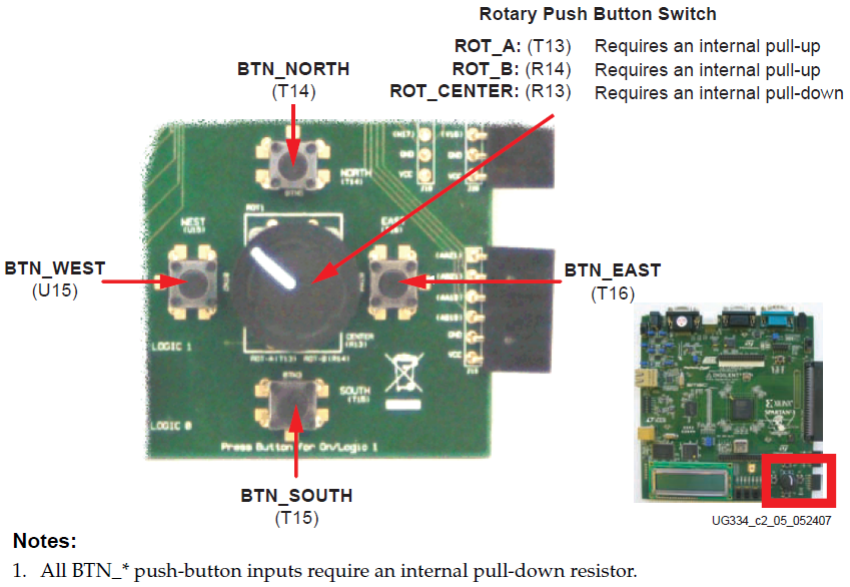


Figura 3.19: Pulsadores configurables [1].

Los pulsadores mostrados en la figura anterior siguen el esquema que se muestra a continuación, donde podemos ver la inclusión de una resistencia interna *pull–down* (figura 3.20) que nos dará un nivel lógico bajo (*low*) cuando no esté presionado dicho pulsador, evitando con ello posibles errores.

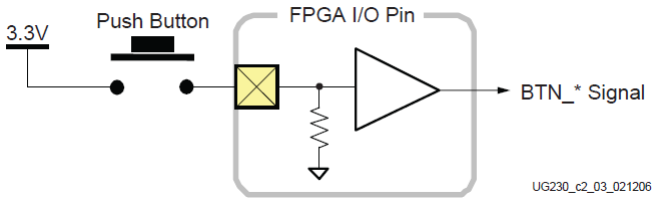


Figura 3.20: Esquema de funcionamiento de un pulsador configurable [1].

A continuación, mostramos la correcta configuración para la conexión a la FPGA de los pulsadores con las resistencias *pull–down* incluidas.

```
1 NET "BTN_EAST" LOC = T16 | IOSTANDARD = LVCMOS33 | PULLDOWN;
2 NET "BTN_NORTH" LOC = T14 | IOSTANDARD = LVCMOS33 | PULLDOWN;
3 NET "BTN_SOUTH" LOC = T15 | IOSTANDARD = LVCMOS33 | PULLDOWN;
4 NET "BTN_WEST" LOC = U15 | IOSTANDARD = LVCMOS33 | PULLDOWN;
```

3.9. Indicadores LED

La placa dispone de indicadores LED tanto configurables como de control.

- **Indicadores LED configurables:** Para la señalización de diferentes señales podremos hacer uso de los siete indicadores LED que contiene para ello la placa con la que se trabaja en este proyecto.

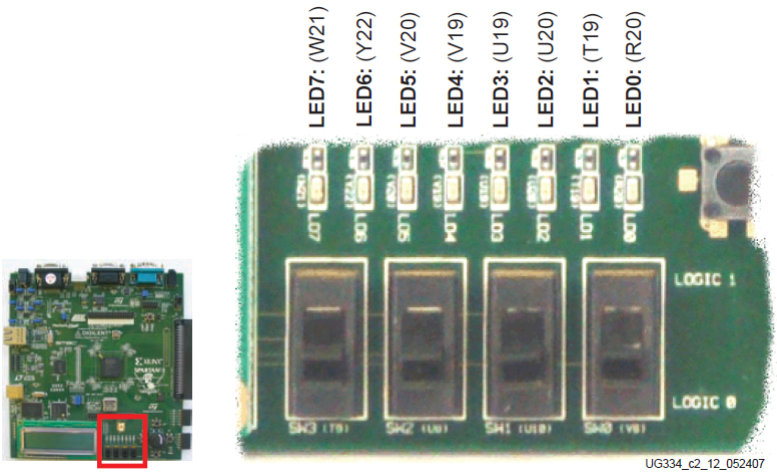


Figura 3.21: Indicadores LED configurables [1].

La asignación de pines en el fichero UCF es el que se ve a continuación:

```
1 NET "LED<7>" LOC = W21 | IOSTANDARD = LVCMOS33 | SLEW = SLOW
      | DRIVE = 8;
2 NET "LED<6>" LOC = Y22 | IOSTANDARD = LVCMOS33 | SLEW = SLOW
      | DRIVE = 8;
3 NET "LED<5>" LOC = V20 | IOSTANDARD = LVCMOS33 | SLEW = SLOW
      | DRIVE = 8;
4 NET "LED<4>" LOC = V19 | IOSTANDARD = LVCMOS33 | SLEW = SLOW
      | DRIVE = 8;
5 NET "LED<3>" LOC = U19 | IOSTANDARD = LVCMOS33 | SLEW = SLOW
      | DRIVE = 8;
```

```
6 NET "LED<2>" LOC = U20 | IOSTANDARD = LVCMOS33 | SLEW = SLOW
      | DRIVE = 8;
7 NET "LED<1>" LOC = T19 | IOSTANDARD = LVCMOS33 | SLEW = SLOW
      | DRIVE = 8;
8 NET "LED<0>" LOC = R20 | IOSTANDARD = LVCMOS33 | SLEW = SLOW
      | DRIVE = 8;
```

- **Indicadores LED de control:** Como se muestra en la siguiente figura, la placa dispone de tres indicadores LED que nos dirán el estado en el que se encuentra la programación.

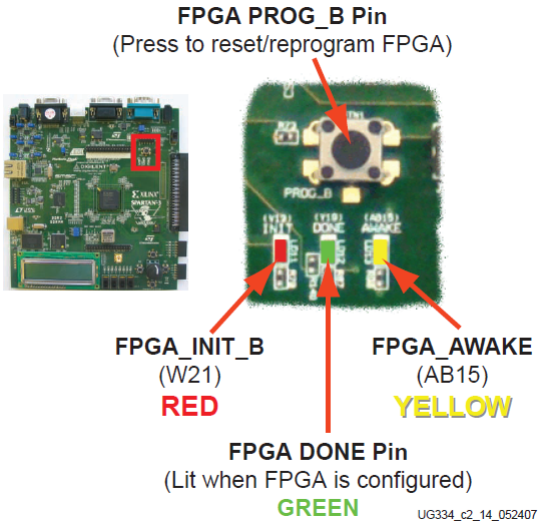


Figura 3.22: Indicadores LED de control de la placa [1].

- *FPGA_INIT_B*: Este LED de color rojo se iluminará principalmente indicándonos lo siguiente:
  - Cuando presionamos el botón *PROG_B* para el reset de la FPGA.
  - Cuando ocurre un error en la programación del código en la FPGA, permaneciendo apagado el LED '*FPGADONE*'.
- *FPGA DONE*: Nos indica con un LED de color verde cuándo la FPGA está correctamente configurada a la hora de programarla.

- $FPGA\_AWAKE$: Nos indica con un LED de color amarillo cuándo el programa que hemos instalado en la FPGA se encuentra en modo suspensión.

Si fuera necesario, se pueden configurar los indicadores LED de control. Para ello deberemos consultar el enlace [1] del anexo *Referencias*.

Capítulo 4

Herramientas Software

4.1. ISE Design Suite

El software en el que configuraremos nuestro código será el *ISE Design Suite 14.7* de *Xilinx, Inc.*, el cual podrá ser descargado de la página oficial del fabricante en su versión de evaluación de 30 días, después de habernos registrado en su *website*. Esta es la última versión del programa *ISE Design Suite*, sustituido por *Vivado Design Suite* para la configuración de las nuevas FPGAs. La utilización del antiguo programa, cuya versión data de 2013, se debe a que las únicas FPGAs *SPARTAN* que pueden ser configuradas con *Vivado* son las *SPARTAN–6*.

4.1.1. Licencia

La licencia del software se obtiene una vez descargado e instalado el programa en la siguiente ruta desde el menú de inicio de Windows®:

Todos los programas > *Xilinx Design Tools* > *ISE Design Suite* 14,7 > *Accessories* > *Manage Xilinx Licenses*

En la pestaña *Acquire a License* seleccionaremos el tipo de licencia que queramos obtener fijándonos en la descripción y las posibles limitaciones de cada una de ellas en el cuadro de texto *Description of the above selected option*. Siguiendo las instrucciones por pantalla, finalmente tendremos el archivo *.lic* que lo almacenaremos, preferiblemente, en

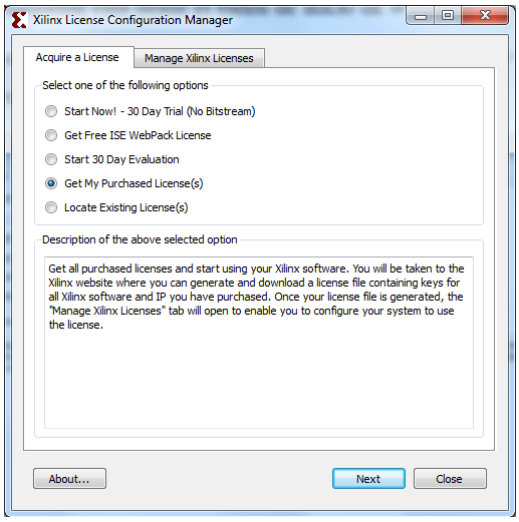


Figura 4.1: Adquirir licencia *ISE Design Suite 14.7. Elaboración propia.*

la carpeta donde hemos instalado el programa.

Una vez hayamos obtenido la licencia del software, deberemos cargarla desde la misma aplicación anterior en la pestaña *Manage Xilinx Licenses* haciendo clic en el botón *Copy License...* y donde tendremos que seleccionar el archivo *.lic* de la ubicación donde hemos guardado la licencia proporcionada por *Xilinx, Inc.*

Una vez cargada la licencia de nuestro producto podremos ver, entre otras cosas, la fecha de expiración de algunas de sus aplicaciones, en caso de tener alguna versión limitada.

4.1.2. Crear Nuevo Proyecto

Abriremos el programa haciendo doble clic en el acceso directo creado en la instalación de manera automática en nuestro escritorio. Para comenzar a elaborar un nuevo proyecto en *ISE Design Suite 14.7*, hacemos clic en *File > New Project...* y tendremos unas pantallas como las que siguen.

En el apartado *Name* insertaremos el nombre de nuestro proyecto como puede ser, por ejemplo, *PFC_Hardware*. Automáticamente, y por defecto, la localización de los archivos del proyecto que vayamos a realizar, así como su directorio de trabajo, estarán contenidos en una carpeta dentro de nuestro disco duro con el nombre que le hemos dado al proyecto.

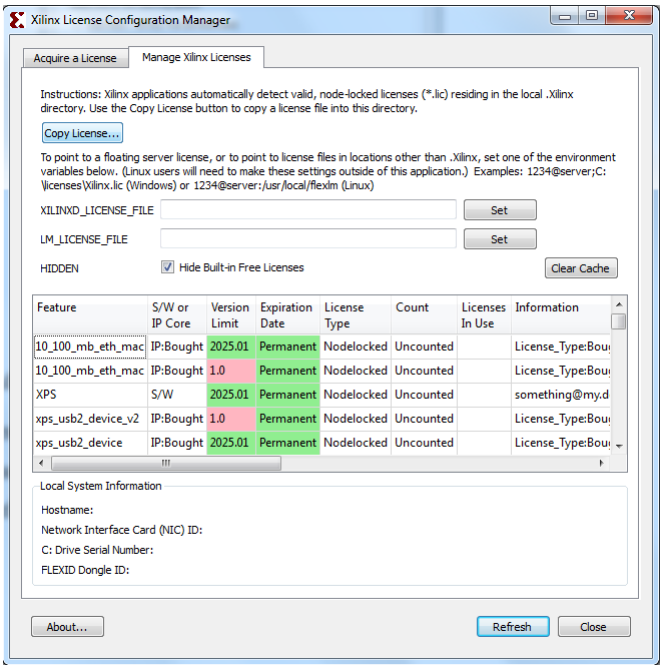


Figura 4.2: Cargar licencia de *ISE Design Suite 14.7*. *Elaboración propia.*

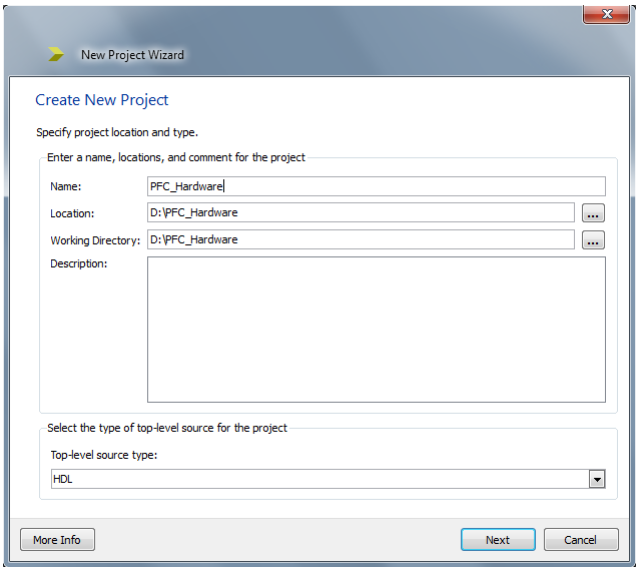


Figura 4.3: Nuevo Proyecto en *ISE Design Suite 14.7*. *Elaboración propia.*

El proyecto será realizado en lenguaje VHDL, por lo que en el desplegable *Top–level source type* indicaremos que el *top–level* es *HDL* o *Hardware Description Lenguage*.

Haciendo clic en *Next*, nos aparecerá la pantalla donde podremos configurar los ajustes del proyecto para indicar las propiedades del mismo.

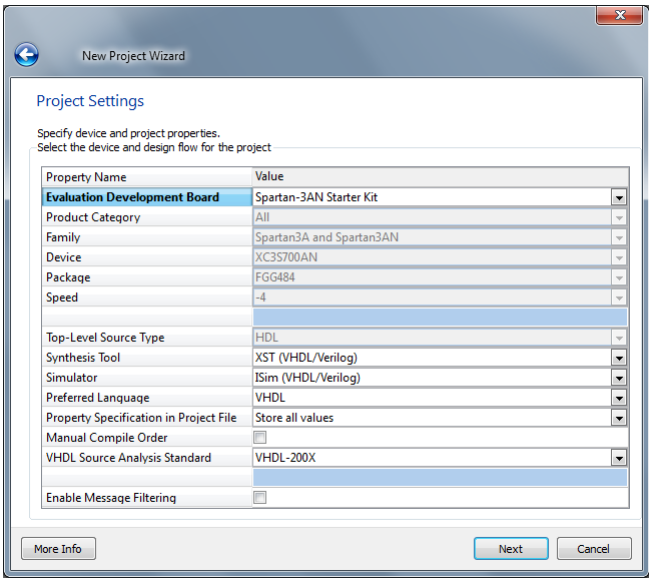


Figura 4.4: Ajustes del Proyecto en *ISE Design Suite 14.7*. *Elaboración propia.*

Indicando que poseemos la placa *Spartan–3AN Starter Kit* se auto–configurarán las opciones para la definición del tipo de FPGA empleada, que son las dadas por el fabricante para la FPGA utilizada. En esta ventana también es posible determinar con qué herramienta podremos simular nuestro proyecto, siendo el de defecto (*ISim*) el elegido debido a que ya se encuentra instalado dentro del *ISE Design Suite 14.7* y para otros como el *Modelsim* o *Questa* es necesario un nuevo software de pago. Otro apartado a tener en cuenta es la versión del lenguaje de programación a utilizar, puesto que en versiones anteriores, como la de 1987, no existen palabras clave que podemos llegar a utilizar en el desarrollo de nuestra aplicación como pueden ser ROL, ROR, SLA, SLL, XNOR entre muchas otras.

En la pantalla siguiente tendremos el resumen de la configuración del proyecto que acabamos de realizar. Haciendo clic en el botón *Finish* terminaremos la configuración de trabajo y podremos proceder a la escritura de los códigos.

4.1.3. Entorno de trabajo

La pantalla inicial del programa después de haberlo configurado para nuestro propósito es la mostrada en la siguiente figura, donde en ella existen algunos puntos importantes

que deberemos conocer.

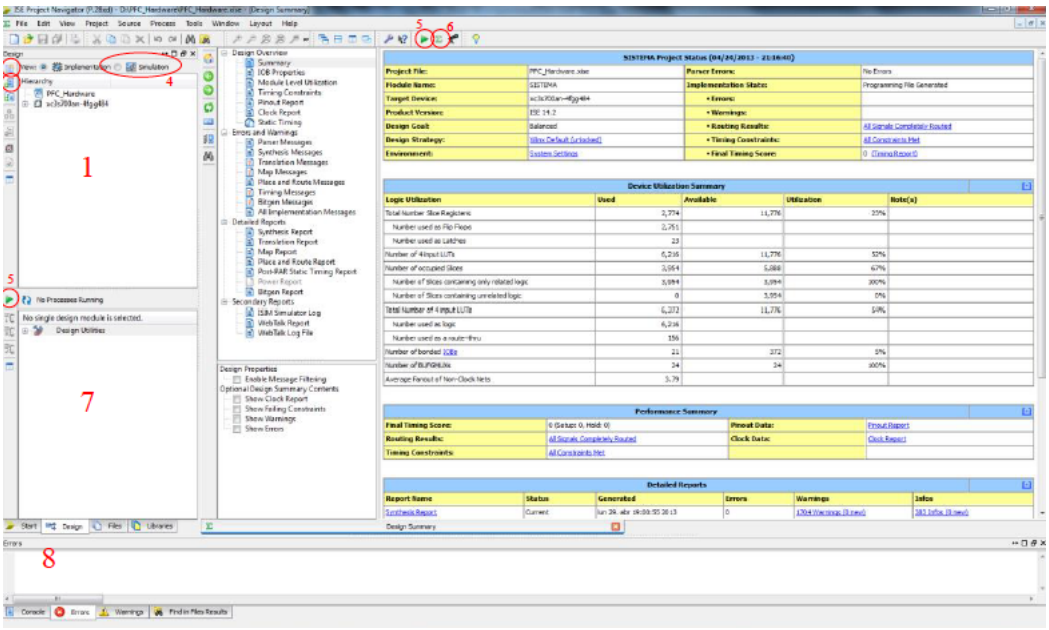


Figura 4.5: Entorno de trabajo del programa *ISE Design Suite 14.7. Elaboración propia.*

1. En esta ventana se nos mostrará la jerarquía del diseño que vayamos implementando.

2. Este punto nos ofrece la posibilidad de añadir una nueva fuente de código a nuestro proyecto y la cuál veremos reflejada en la ventana de la jerarquía de diseño.

3. Es posible añadir archivos ya creados anteriormente si hacemos clic en esta opción.

4. Haremos clic sobre él si estamos editando los archivos fuente destinados a la simulación del proyecto.

5. Síntesis e implementación del proyecto. Una vez realicemos esta operación, podremos depurar el código en caso de errores de síntesis e/o implementación.

6. En todo momento, y una vez realizada la síntesis e implementación del proyecto, podremos ver el resumen del diseño del mismo en *Design Summary*, que es la ventana abierta que se muestra en la figura 4.5.

7. Herramientas de síntesis y mapeado del proceso seleccionado en el punto 1 de la lista actual.

8. Consola que nos mostrará los errores y *warnings* existentes en el código de los archivos.

4.1.4. Nuevo archivo fuente

Una vez hayamos preparado el entorno de trabajo, procederemos a incluir los archivos fuente donde escribiremos el código de cada una de las partes de las que se compone el proyecto. Para ello, y situándonos en la ventana principal del programa, pulsamos el botón indicado en la figura 4.5, punto 2, *New Source*.

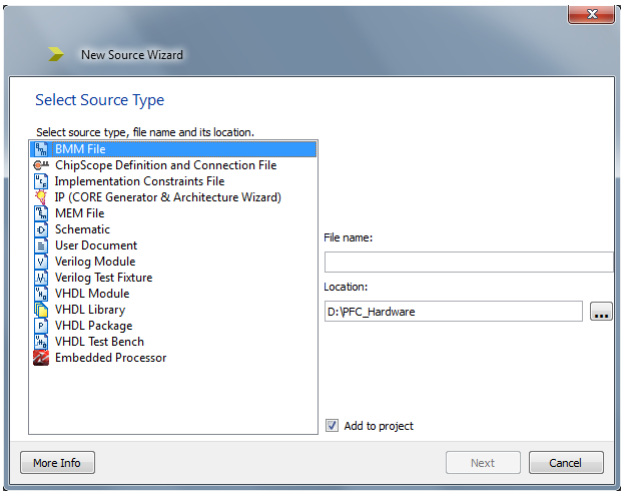


Figura 4.6: Agregar nuevo archivo fuente a un proyecto *ISE Design Suite 14.7. Elaboración propia.*

En el menú que nos aparecerá deberemos elegir el tipo de archivo que nos interese. En *File name* le asignaremos el nombre al archivo y una vez dado el nombre deberemos cerciorarnos de que la casilla de verificación *Add to Project* esté activada para que el archivo se incluya a la jerarquía de nuestro proyecto. Los tipos de archivo que se usarán en este proyecto son los siguientes:

- **Implementation Constraints File:** Archivo con extensión *.ucf* donde indicamos

al programa las conexiones de entrada/salida entre la FPGA y los periféricos que necesitemos. La conexión de cada uno de los pines de la FPGA con sus periféricos se encuentra en la guía de usuario de la placa [1].

- **VHDL Package:** Será el archivo, con extensión *.vhd*, que contendrá todos los tipos, funciones y constantes comunes para todo el proyecto. Al igual que ocurre con el *.ucf*, la pantalla tras pulsar *Next* es un resumen del archivo creado. Deberemos declarar el paquete en todos los *VHDL Module* donde queramos usar sus variables de la siguiente manera:

```
1 use work.nombre_del_paquete.all;
```

- **VHDL Module:** Con extensión *.vhd*, es el archivo donde escribiremos nuestro código en VHDL. En la pantalla siguiente podremos incluir, de forma opcional, el nombre del puerto en el módulo (figura 4.7) así como su dirección (in/out/inout) o incluso, si se trata de un bus, activando la casilla correspondiente para ello indicando el tamaño del mismo en las columnas MSB o *Most Significant Bit* ?-LSB o *Least Significant Bit*, o si lo preferimos, definirlos cuando estemos editando el archivo.

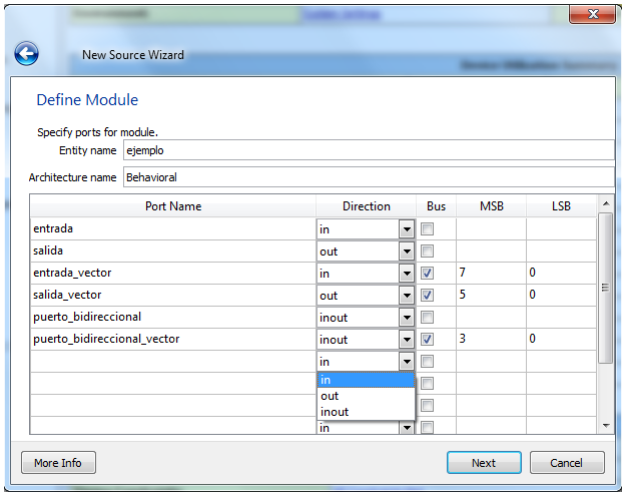


Figura 4.7: Declaración de puertos en un módulo VHDL. *Elaboración propia.*

La pantalla siguiente al editor de puertos es el resumen del archivo que acabamos de crear:

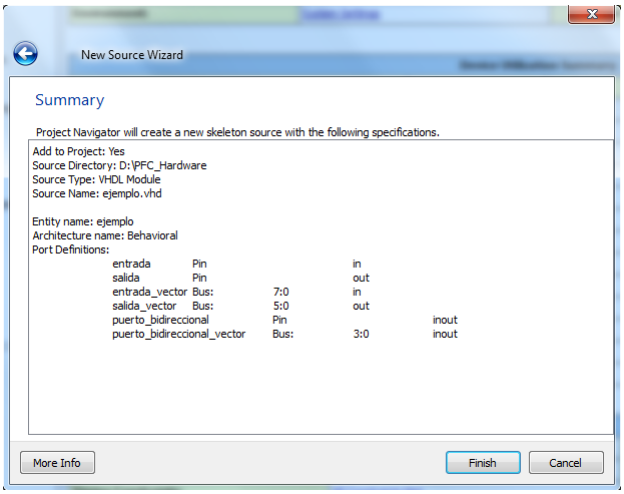


Figura 4.8: Resumen de los puertos de un módulo. *Elaboración propia.*

Finalmente, al presionar sobre el botón *Finish* de esta última pantalla, tendremos el siguiente código generado a través de las especificaciones dadas:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  --Declaración de la librería creada en un VHDL Package
4  --use work.nombre_del_paquete.all;
5  entity ejemplo is
6  Port ( entrada : in STD_LOGIC;
7  salida : out STD_LOGIC;
8  entrada_vector : in STD_LOGIC_VECTOR (7 downto 0);
9  salida_vector : out STD_LOGIC_VECTOR (5 downto 0);
10 puerto_bidireccional : inout STD_LOGIC;
11 puerto_bidireccional_vector : inout STD_LOGIC_VECTOR (3
      downto 0));
12 end ejemplo;
13 architecture Behavioral of ejemplo is
14 --Declaración de señales para este módulo
```

```
15 begin
16 --Declaración de variables para este módulo
17 --Instrucciones de funcionamiento del módulo (concurrente,
      serie o mixto)
18 end Behavioral;
```

- **VHDL Test Bench:** Es el archivo que se genera de manera automática para la simulación del bloque que queramos depurar. Cuando se comente la parte de simulación de alguno de los códigos realizados se describirá más ampliamente este apartado.

4.1.5. Síntesis y mapeado

Siempre que se cree o modifique algún archivo *.vhd* de la vista *Implementation* (figura 4.9) lleva consigo el tener que sintetizar y mapear todo el diseño, a excepción de los archivos *.ucf* cuyos cambios en estos no implica una nueva síntesis del diseño pero sí un nuevo mapeado, debido a que se trata de un archivo que se genera para 'conectar' los pines de la FPGA a los distintos periféricos y no supone un cambio en el código del programa. En la parte superior de la figura 4.9 podemos encontrar la jerarquía (*Hierarchy*) del proyecto, mientras que en la parte inferior (*Processes*) es donde realizaremos la síntesis y mapeado del mismo, la cual sólo estará disponible si el archivo seleccionado en *Hierarchy* es el establecido como *Top Module* o archivo principal del programa. La asignación como *Top Module* a un determinado archivo *.vhd* se realiza haciendo clic con el botón propiedades de nuestro teclado o ratón sobre un archivo y seleccionando la opción *Set as Top Module*. En la ventana de procesos (*Processes*) podemos diferenciar algunos apartados como son:

- **Design Utilities:** Donde podremos generar el esquema del bloque seleccionado en *Hierarchy*. Este apartado, al igual que *Check Syntax*, son los únicos apartados comunes a todos los archivos.

- **Synthesize–XST:** Analizaremos la sintaxis del código y generaremos el *RTL Schematic*. Los posibles errores o *warnings* se verán reflejados en la consola del programa.

- **Implement Design:** Será donde se realice el mapeado y conexionado (*Place and Route*) del proyecto. Este apartado realizará la síntesis del código en caso de no haberse hecho anteriormente. Los posibles errores o *warnings* se verán reflejados en la consola del programa.

- **Generate Programming File:** Si no existe ningún error, se creará un archivo con extensión *.bit* que contendrá el programa que hayamos realizado y que será el que posteriormente podamos instalar en la FPGA. El nombre de este archivo será el mismo que el del archivo *.vhd* que ejerce como *Top Module* del cual se ha generado.

- **Configure Target Device:** Una vez implementado el diseño, esta opción abrirá un nuevo programa, el *iMPACT*, para la programación física de la tarjeta *Spartan–3AN* de *Xilinx, Inc.*

4.1.6. Simulación

Anteriormente, se comentó la existencia de un tipo de archivo *.vhd* creado de manera automática para la simulación del código seleccionado y que recibe el nombre de *VHDL Test Bench*. Dicha selección puede hacerse en la ventana posterior a la elección del archivo para la simulación, mostrada en la figura 2.33.

Al elegir el archivo *Top Module*, en nuestro caso llamado 'SISTEMA', se creerá el código para la simulación de todo el proyecto. Este archivo creado de manera automática será el empleado por el *ISim* para la simulación del código. Haciendo doble clic en *Simulate Behavioral Model* en la ventana *Processes* ejecutaremos el programa de simulación *ISim* con el archivo creado anteriormente.

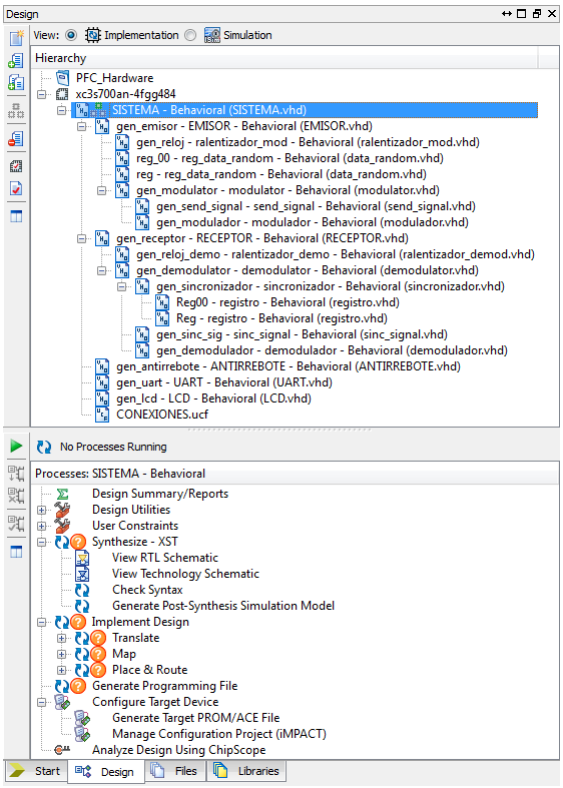


Figura 4.9: Ventana de diseño (*Implementation*) en *ISE Design Suite 14.7*. *Elaboración propia.*

4.2. ISim

El *ISim 14.7* es el software donde realizaremos todas las simulaciones de los códigos que vayamos generando para comprobar el correcto funcionamiento de estos antes de la implementación en la tarjeta de manera física. La interfaz del programa se muestra en la siguiente figura.

Entre los puntos a destacar en este software se encuentran los siguientes:

1. **Instance and Process Name:** Es donde se encuentran albergados todos los bloques del proyecto que hemos simulado.

2. **Objects:** Seleccionando alguno de los bloques que hay en *Instance and Process Name*, aquí podremos ver las señales existentes en él.

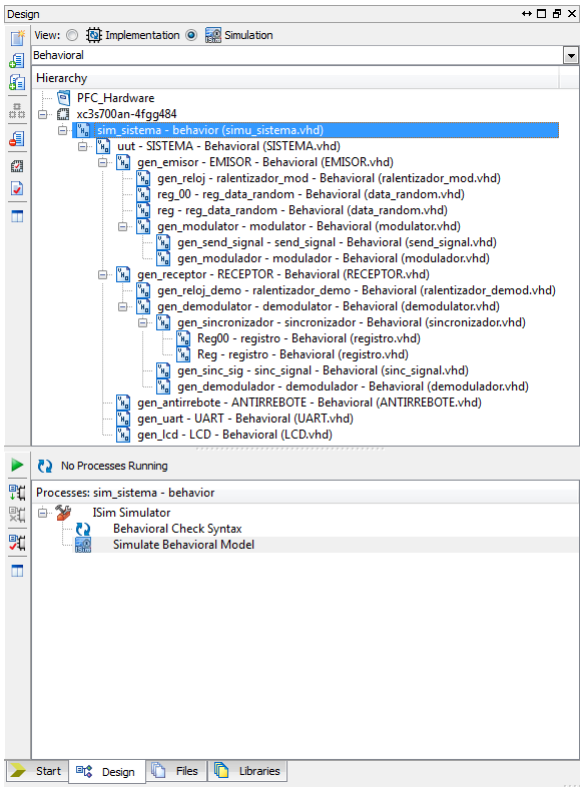


Figura 4.10: Ventana de diseño (*Simulation*) en *ISE Design Suite 14.7. Elaboración propia.*

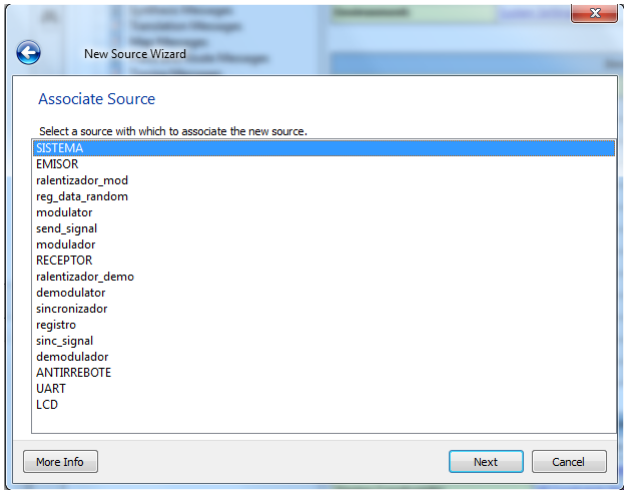


Figura 4.11: Selección del archivo que se pretende simular. *Elaboración propia.*

3. **Name:** Es el lugar al cual arrastramos las señales ubicadas en el apartado *Objects*.

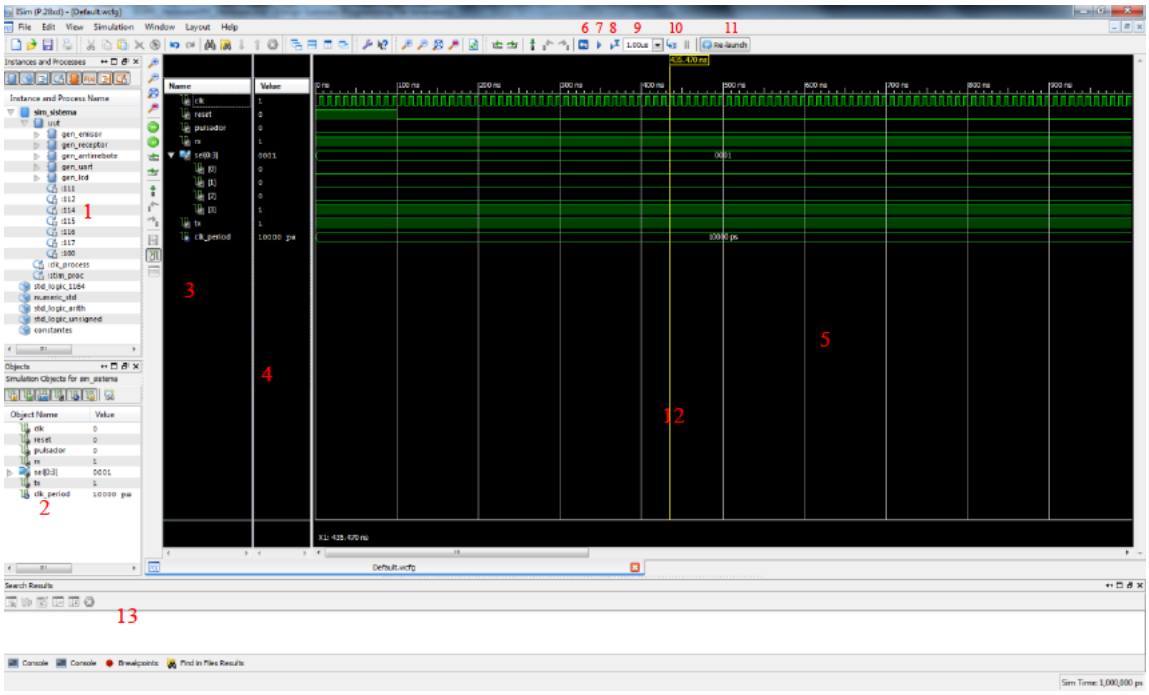


Figura 4.12: Interfaz *ISim 14.7*. *Elaboración propia*.

4. **Value:** Valor de la señal existente en la columna denominada *Name* en cada instante marcada por la línea de tiempo de color amarilla mostrada en la figura 4.12.

5. **Seguimiento de la simulación:** Es donde veremos reflejadas las señales seleccionadas en el tiempo de simulación.

6. **Restart:** Pulsaremos este botón cuando queramos reiniciar la simulación desde el tiempo cero.

7. **Run All:** Al pulsar sobre este botón correrá la simulación hasta que la detengamos pulsando *Break* (10).

8. **Run for the time specified on the toolbar:** Presionando este botón simularemos nuestro proyecto por el tiempo especificado en la barra de herramientas (9).

9. **Specific time:** Determinaremos un tiempo específico para la simulación.

10. **Break:** Nos sirve para parar la simulación cuando hemos actuado sobre el botón *Run All* (7).

11. **Re–launch:** Cuando hemos modificado algo en el código de nuestro programa en *ISE Design Suite*, no es necesario ejecutar de nuevo desde este último la simulación, sino que pulsando este botón actualizaremos el proyecto y podremos seguir simulándolo.

12. **Markers:** Son marcadores de tiempo para medir, por ejemplo, el tiempo de respuesta de algunas señales. Es posible añadir tantos marcadores de tiempo como queramos haciendo clic con el botón de propiedades sobre la pantalla de seguimiento de la simulación (5) y yendo a $Markers \rightarrow AddMarker$.

13. **Console:** Consola del programa donde podremos ver el estado de la simulación así como errores y *warnings* existentes en ella (estos errores o *warnings* que puedan haber vendrán a consecuencia de la modificación de manera errónea del código una vez abierto el *ISim* y pulsemos *Re–launch* (11)).

4.3. ISE iMPACT

Es el software mediante el cual programaremos nuestro proyecto en la placa *Xilinx Spartan-3AN Starter kit*. Haciendo doble clic en *Configure Target Device* dentro de la ventana *Processes* del *ISE Design Suite 14.7* (figura 4.9), lanzaremos este software. Si no existen archivos del programa *ISE iMPACT* dentro de nuestro proyecto, aparecerá un *warning* avisándonos de ello (figura 4.13). Pulsando OK accederemos al programa.

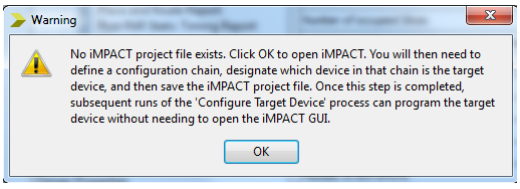


Figura 4.13: *Warning* debido a la no existencia de archivos de *iMPACT* en el proyecto. *Elaboración propia.*

4.3.1. Comunicación PC–Board

La comunicación entre la placa y el software se realizará mediante cable USB tal como muestra la siguiente figura.

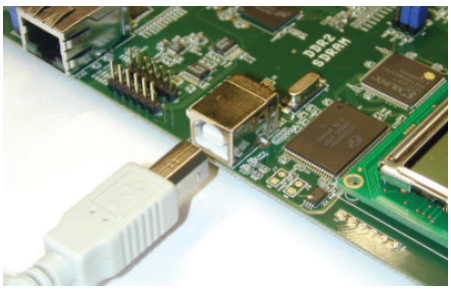


Figura 4.14: Conexión USB–Placa [1].

Al conectar por vez primera la placa al PC se autoinstalarán los drivers necesarios para la comunicación entre ambos. A través de dos indicadores LED podremos saber el estado de la conexión USB–placa, siendo **verde** cuando esté lista y preparada la comunicación, o **rojo** cuando la placa esté descargando e instalando actualizaciones para sí misma.

4.3.2. Configuración de la FPGA

La primera ventana del programa *ISE iMPACT* está vacía, por lo que iremos a $File \rightarrow NewProject$. Inmediatamente después de realizar esta acción, el software nos preguntará si queremos realizar la implementación del proyecto de manera automática, a lo cual responderemos según nos convenga, que en nuestro caso es *Yes*. Esta elección dará como resultado la siguiente pantalla:

En esta última pantalla elegiremos la opción por defecto que es la que se muestra en la figura anterior, y posteriormente presionando *OK* el programa comprobará la conexión del cable de programación a la placa. En caso erróneo, se nos mostrará un mensaje de error informándonos de la imposibilidad de conectarse a la placa.

Acto seguido, siempre y cuando la conexión entre el PC y la placa sea la correcta, el software nos mostrará un cuadro de texto por si queremos cargar en ese mismo momento

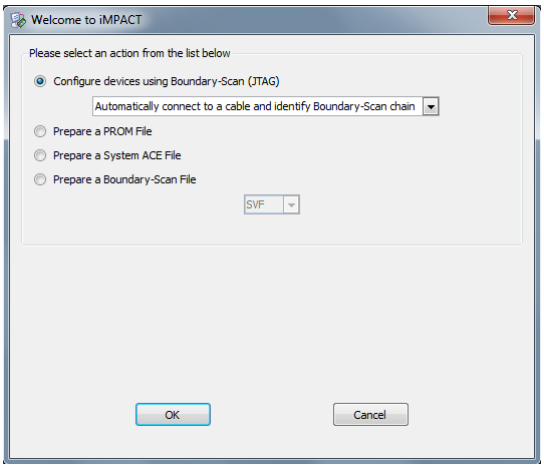


Figura 4.15: Autoconfiguración de la implementación del proyecto. *Elaboración propia.*

el archivo *.bit* que contiene el proyecto realizado en el *ISE Design Suite* (figura 4.16). Este archivo *.bit* es creado al ejecutar la opción de *Generate Programming File* como ya hemos visto anteriormente en el apartado 4.1.5.

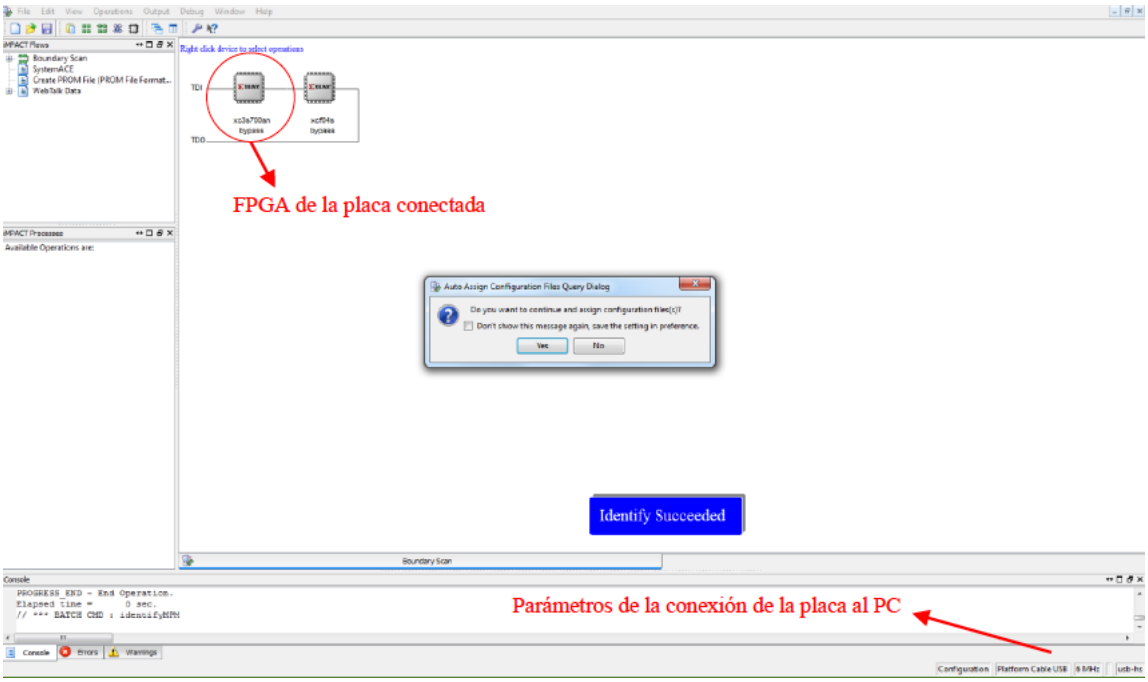


Figura 4.16: Autoconfiguración de la implementación del proyecto. *Elaboración propia.*

En el caso de responder *Yes*, que será el caso que nos ocupa, tendremos que seleccionar el archivo *.bit* que queramos cargar en la FPGA (figura 4.17).

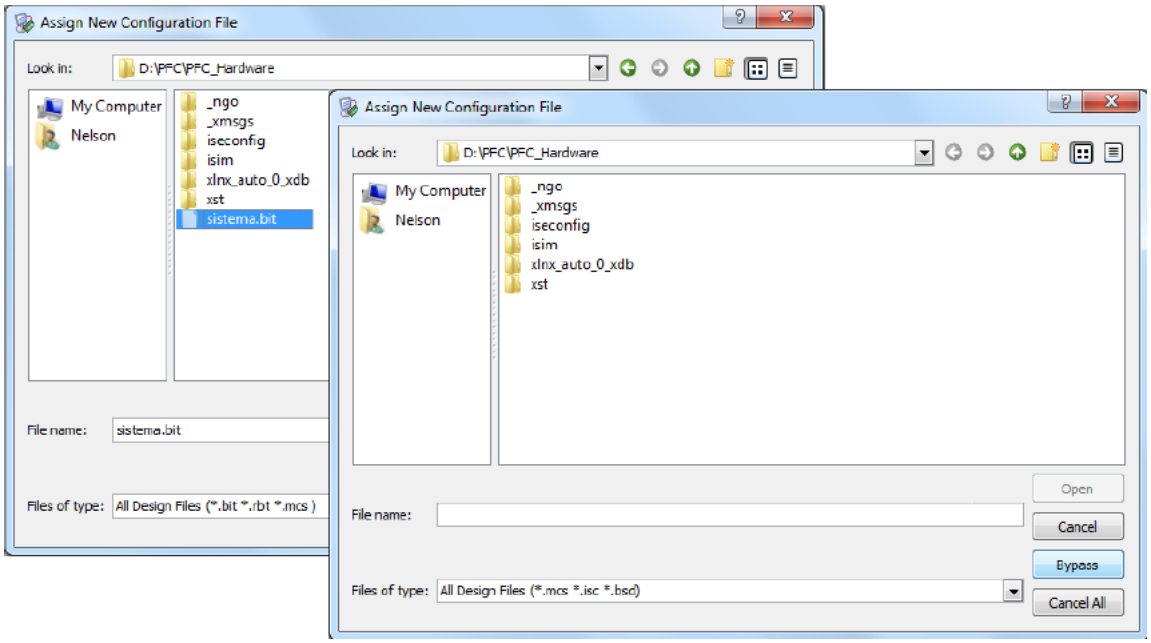


Figura 4.17: Cargando archivos de configuración en la FPGA. *Elaboración propia.*

En la primera pantalla seleccionaremos el archivo *.bit* antes descrito y posteriormente escogeremos la opción de *bypass* en la elección del archivo de memoria, debido a que en este proyecto no se ha usado para guardar el programa en la placa aún habiendo cortado la alimentación de la misma. Finalmente, se nos mostrará una pantalla resumen de lo hecho a la cual daremos el *OK*.

4.3.3. Programación de la FPGA

El último paso para la programación de la FPGA será haciendo clic en la opción *Program FPGA Only* dando como resultado la pantalla que se muestra a continuación si todo se ha ejecutado de manera correcta.

A partir de este momento, ya podremos realizar las pertinentes pruebas del programa realizado teniendo en cuenta que, si la placa pierde la alimentación de corriente, se perderán todos los datos instalados en la FPGA, debido a que no se ha usado una memoria para guardarlos.

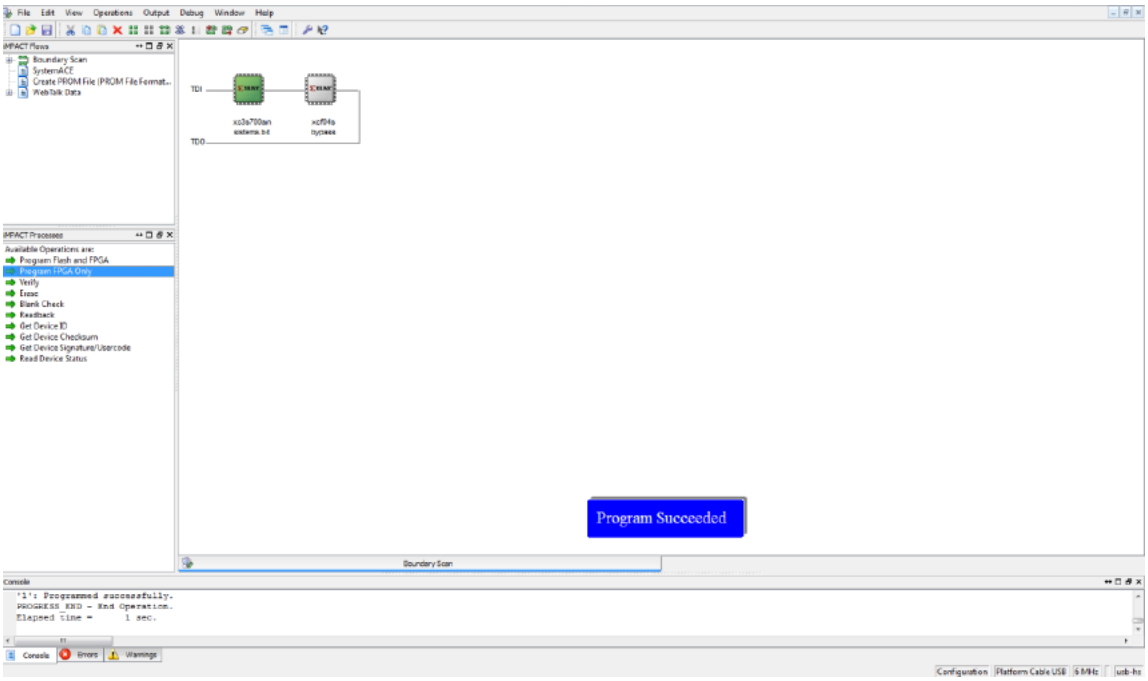


Figura 4.18: Programación satisfactoria de la FPGA. *Elaboración propia.*

4.4. Qt

Figura 4.19: *Framework* multiplataforma orientado a objetos. *Qt Company* [2].

Qt en su versión 5.8 es la herramienta de programación multiplataforma que se ha usado sobre *Windows 10*® para la realización del software necesario para el tratamiento de los datos entre el PC y la FPGA programada. Este entorno de desarrollo consiste en

un conjunto de herramientas de desarrollo y una interfaz gráfica, el *Qt Designer*, que gobierna el proceso de desarrollo. En el diseño gráfico, editamos las ventanas y añadimos botones, menús desplegables, barras de progreso y otros objetos de interfaz gráfica. La programación de las acciones asociadas a cada elemento de la interfaz se realiza mediante lenguaje *C++*, teniendo acceso a los recursos de los elementos gráficos pudiendo alterar su salida o entrada. *Qt Creator* permite conseguir que el entorno visual del programa creado sea igual que cualquier programa usado en un entorno de *Windows*®.

También, cabe destacar las librerías que ayudan al proyecto en cualquier tipo de entorno que tengamos de partida así como comunicaciones de diferentes niveles, la capacidad de hacer gráficos o incluso la de leer en otros formatos, por ejemplo, *HTML* o *XML*.

El software en el que programaremos nuestro entorno en *Windows 10*®, el *QT Creator 4.2.1*, podrá ser descargado de su página oficial ubicada en el siguiente enlace del anexo de *Referencias* [2]. **Al ser un proyecto de código abierto, su licencia es gratuita**.

4.4.1. Crear Nuevo Proyecto

El programa se abre haciendo doble clic en el acceso directo creado en la instalación de la aplicación en nuestro escritorio. Para comenzar a elaborar el nuevo proyecto en *Qt Creator*, haremos clic en $File \rightarrow New\ File\ or\ Project...$ y tendremos la siguiente pantalla:

A continuación, elegiremos la plantilla *Qt Widgets Application* para comenzar con el diseño de la interfaz gráfica que queremos realizar para el proyecto actual, insertando antes el nombre del mismo, que será *SGandO*, en referencia a las siglas *Signal Generator and Oscilloscope*, y que describen las funciones para el proyecto que se podrán manipular y ver a través de la interfaz gráfica. Automáticamente, tendremos el directorio de trabajo con el nombre que le hemos dado al proyecto donde estarán todos los ficheros que crearemos. Pulsando sobre el botón *Next*, seleccionaremos todos los kits haciendo clic en la pestaña *Select all kits* para, haciendo nuevamente clic en *Next*, definir la información

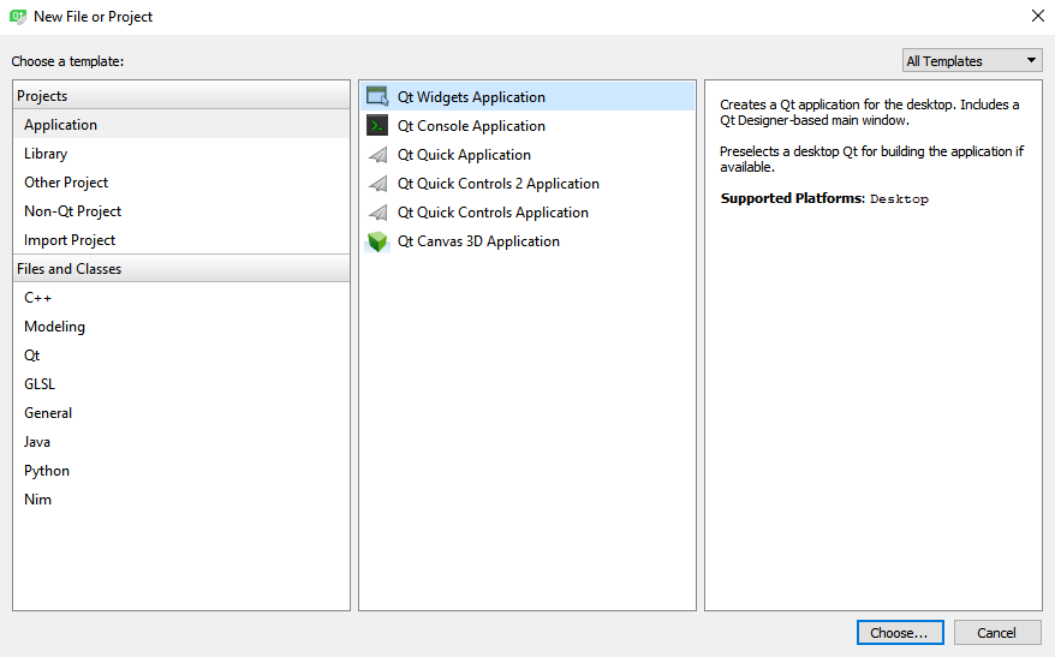


Figura 4.20: Crear nuevo proyecto en *Qt Creator*. *Elaboración propia.*

básica para los ficheros que crearemos.

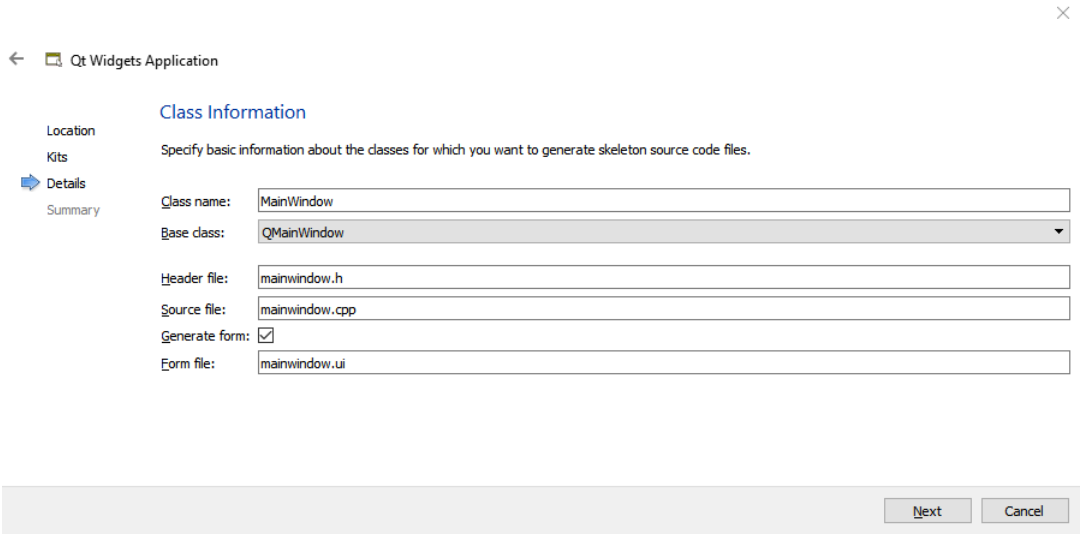


Figura 4.21: Información básica para los ficheros en *Qt Creator*. *Elaboración propia.*

Por último, se nos mostrará un resumen de los ficheros que hemos creado, a lo cual haremos clic en el botón *Finish* para comenzar con el diseño de lo que será la interfaz gráfica del proyecto.

4.4.2. Añadir nuevos archivos

Una vez hayamos preparado el entorno de trabajo y teniendo ya la ventana principal, podremos incluir más archivos donde escribiremos el código de cada una de las partes de las que se compone el proyecto. Para ello, y situándonos en la ventana principal del programa, como se mostró en la figura 4.20, iremos a la sección *Files and Classes* para hacer clic en $Qt \rightarrow C++ Class$ (figura 4.22). En esta nueva ventana, únicamente daremos nombre a la *Clase*, que será el mismo para el fichero de *Cabecera* o *Header* y para el fichero *Fuente* o *Source*, sin incluir ninguno de los ítems propuestos.

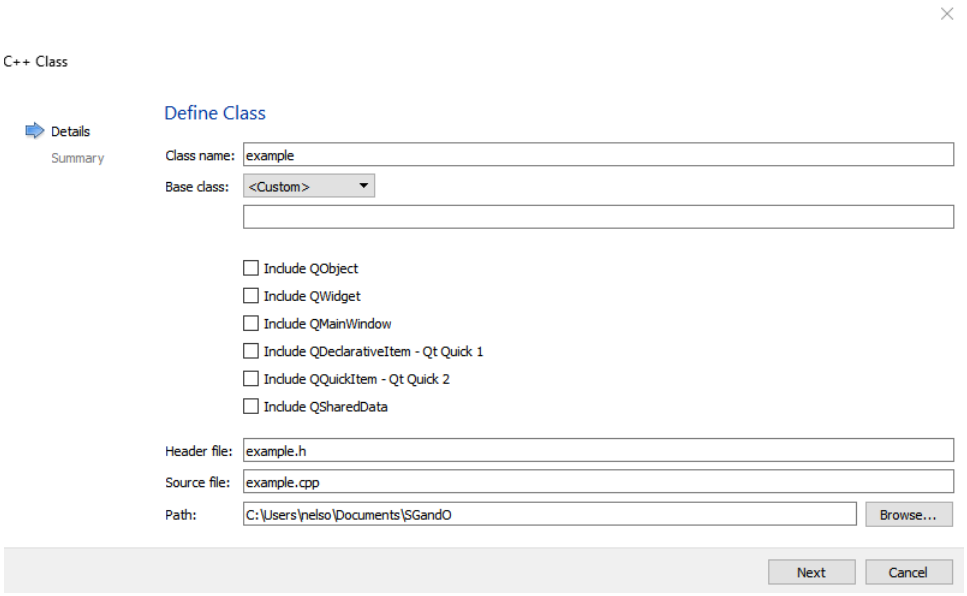


Figura 4.22: Creación de nueva clase. *Elaboración propia.*

Qt Creator clasifica el código en cuatro tipos de ficheros dentro de *Project*: formularios o *forms*, ficheros de cabecera o *headers*, ficheros de código fuente o *sources* y ficheros de recursos o *resources*, los cuales son descritos en los siguientes apartados.

4.4.3. Fichero *Project*

Serán los que posean la extensión **.pro*. Es un fichero que generaremos automáticamente con el *Qt Creator* al empezar el proyecto, cuyo nombre coincidirá con el de la

carpeta creada, en nuestro caso, *SGandO*. Por lo tanto, el fichero del proyecto realizado tomará el nombre *SGandO.pro*.

4.4.4. Ficheros *Forms*

Con extensión **.ui*, este fichero se genera automáticamente al crear una nueva clase desde la ventana de nuevo proyecto, tal como se vio en el apartado 4.4.1. Cada fichero describe en lenguaje *XML* la interfaz gráfica de usuario que usaremos en la aplicación. Las ventanas de la aplicación no tienen por que estar en un fichero **.ui*, ya que existe un conjunto de cuadros de diálogo estándar que se pueden llamar directamente desde el código fuente. Los ficheros **.ui* son diseñados gráficamente en el *Qt Designer*, el cuál genera automáticamente el código *XML* de dichos ficheros. De forma manual, se pueden añadir botones, acciones, gráficos, menús del programa, manejar la posición de la ventana, etc., así como asociar las acciones necesarias a cada elemento gráfico en el fichero de código fuente.

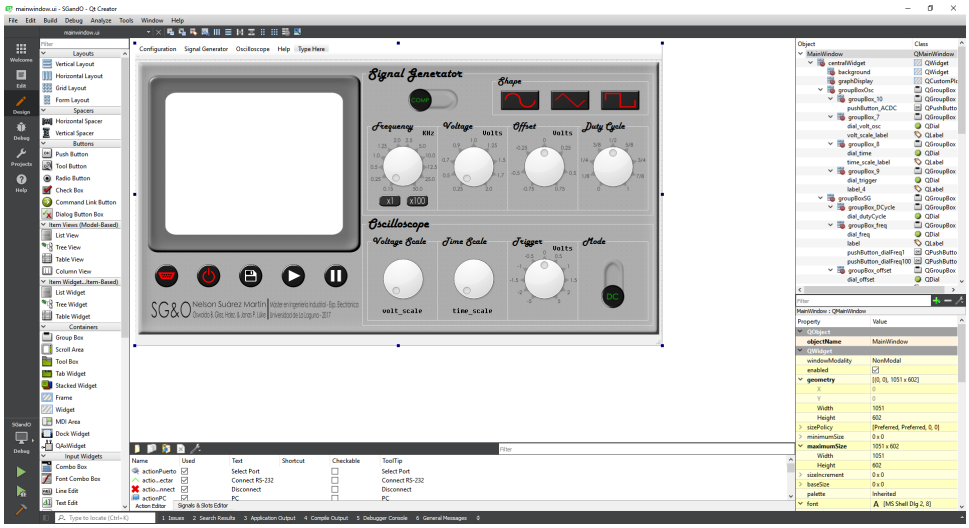


Figura 4.23: Diseño de la interfaz gráfica en *Qt Designer*. *Elaboración propia*.

En caso de necesitar crear nuevas ventanas gráficas, seleccionaremos dentro de la sección *Files and Classes* del menú *New File and Proyect...*, *Qt Designer Form Class*. Esto nos generará tres ficheros con el mismo nombre y diferente extensión como, por

ejemplo, *portwindow.h*, *portwindow.cpp* y *portwindow.ui*.

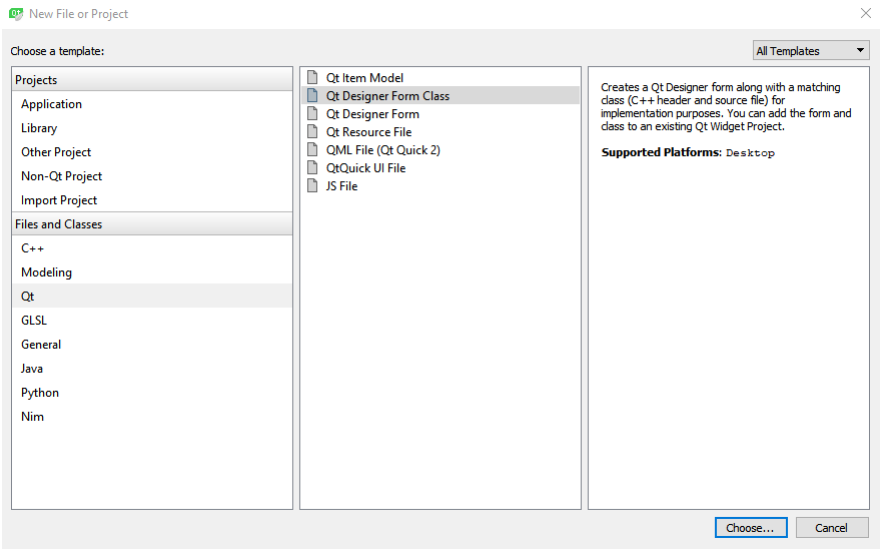


Figura 4.24: Generar una nueva ventana gráfica. *Elaboración propia.*

4.4.5. Ficheros *Headers*

Con extensión **.h*, cada clase que vayamos a crear tendrá un fichero cabecera, por ello, que reciban el mismo nombre. Así, por ejemplo, si la clase a desarrollar se llama *portwindow*, el fichero cabecera tendrá el nombre de *portwindow.h*. Aquí se incluyen, con la directiva *#include*, las cabeceras de las clases que vayan a ser usadas. También es posible definir métodos o atributos, como pueden ser públicos, protegidos o privados. Además de estas categorías, propias de *C++*, en *Qt* se añaden dos categorías adicionales que a su vez pueden ser públicas, protegidas o privadas, que son las *señales* y los *slots*. Las *señales* se emiten cuando en la aplicación se produce un determinado evento y se les puede asociar uno o más *slots* que implementan las acciones correspondientes a esas señales. A continuación, se muestra un ejemplo de fichero de cabecera.

```
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3 #include <QMainWindow>
4
```

```
5  namespace Ui {
6  class MainWindow;
7  }
8  class MainWindow : public QMainWindow
9  {
10 Q_OBJECT
11
12 public:
13 explicit MainWindow(QWidget *parent = 0);
14 ~MainWindow();
15
16 private:
17 Ui::MainWindow *ui;
18 };
19 #endif // MAINWINDOW_H
```

Dentro de las *Cabeceras* del programa también se incluye el archivo donde se definirán las *constantes* del resto de ficheros que componen el proyecto y que no varía mucho de los demás ficheros *Cabecera*. En el proyecto actual, este fichero lleva como nombre *parametros.h*.

```
1  #ifndef PARAMETROS_H
2  #define PARAMETROS_H
3  #include <QtSerialPort/QSerialPort>
4
5  const int MEMO_DISPLAY = 256;
6  const double V_max = 1.65;
7  const int M = 4096;
8
9  // COMUNICACIÓN SERIE RS232 (El número del puerto se selecciona
      por pantalla).
```

```
10 const int baudios = 38400;
11
12 #endif // PARAMETROS_H
```

Para añadir únicamente ficheros *Cabecera*, podremos hacerlo seleccionando *C++ Header File* a través del menú *New File and Proyect...*.

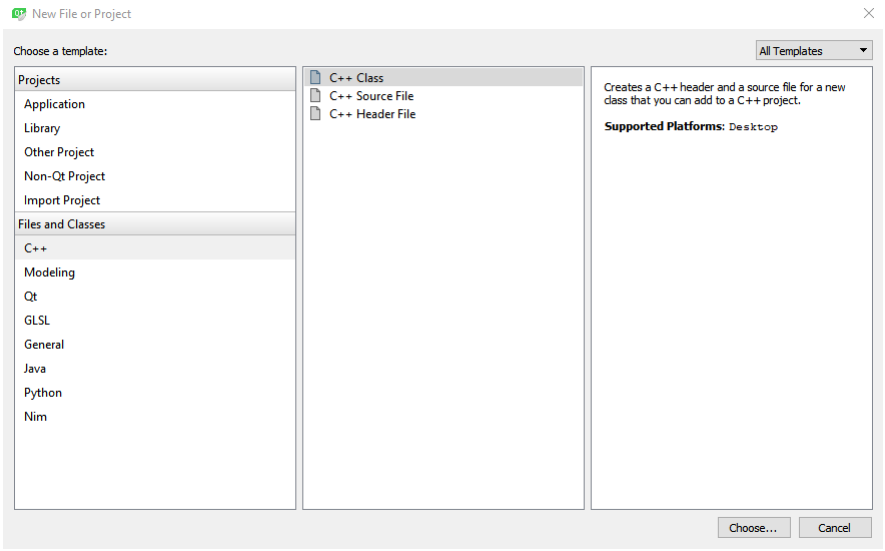


Figura 4.25: Creación de una nueva *Clase*. *Elaboración propia*.

4.4.6. Ficheros *Sources*

Con la extensión **.cpp*, la implementación de cada miembro de una clase está en un fichero con el mismo nombre que la clase que implementa. Así, tal como ocurre con los ficheros de *Cabecera*, con la clase *portwindow*, el fichero tendrá de nombre *portwindow.cpp*. En estos ficheros es donde se implementan las funcionalidades especificadas en la definición de la clase.

A excepción de otros ficheros, en todo proyecto en *Qt* se requiere un fichero principal *main.cpp* que no tendrá fichero de *Cabecera*, que será el punto de inicio del programa a ejecutar. Así mismo, y tal como ocurre en el apartado anterior, podremos incluir ficheros *Fuente* solamente, seleccionando en este caso la opción de *C++ Source File*. A

continuación, podemos ver un ejemplo sobre un fichero *Fuente*.

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
QMainWindow(parent),
ui(new Ui::MainWindow)
{
ui->setupUi(this);
}

MainWindow::~MainWindow()
{
delete ui;
}
```

4.4.7. Ficheros *Resources*

El fichero de recursos tendrá la extensión **.qrc*. Este fichero se añade cuando se incluyen imágenes para iconos y/o fondos en el proyecto. Esto permite incluir las imágenes en el ejecutable, para que en el momento de empaquetar la aplicación, no se pierda la asociación entre estas imágenes y el propio ejecutable.

Para incluir un fichero de este tipo, deberemos seleccionar en la figura 4.24 la opción *Qt Resource File*. Después de darle un nombre y mostrarnos un resumen de lo realizado en esta acción, podremos añadir las imágenes que queramos a través del botón *Add* en la pantalla que nos aparecerá.

4.5. Inkscape

Inkscape es un software libre de diseño de gráficos vectoriales utilizado en este proyecto para realizar los iconos de la interfaz gráfica y el cual puede ser descargado desde el siguiente enlace del anexo de *Referencias* [4].

Figura 4.26: Software de diseño de gráficos vectoriales *Inkscape*. *Inkscape*.

El formato principal de imagen que utiliza es el SVG [15], un formato de gráficos vectoriales bidimensionales en formato XML. Los objetos que vayamos creando pueden ser movidos, rotados, escalados, etc. También es posible ajustar ángulos así como tramas o guías, pudiéndose agrupar y desagrupar diferentes partes al antojo del diseñador.

Entre los diferentes estilos dentro de *Inkscape*, podemos destacar las transparencias que podemos aplicar al objeto, múltiples colores, así como poder llegar a esculpirlo cual escultura se tratase. Un amplio abanico de posibilidades que ha hecho posible realizar los iconos así como el *background* de la interfaz gráfica realizada en *Qt*, y todo ello de manera gratuita y libre. En el *website* de *Inkscape* podremos encontrar vídeos y tutoriales para sacarle el máximo partido a esta herramienta.

Los objetos creados con *Inkscape* son los mostrados en el anexo destinado a ello.

Figura 4.27: Software de diseño electrónico *KiCad*. *KiCad*.

4.6. KiCad

KiCad es un software libre para la creación de esquemas electrónicos, placas de circuito impreso, así como toda la simbología necesaria para ello. Las principales funciones de las que dispone este programa son las ejecutadas haciendo clic en los iconos de la siguiente imagen.

Figura 4.28: Funciones en *KiCad*. *KiCad*.

De izquierda a derecha, el primer icono da acceso al editor de esquemas donde plasmaremos el diseño requerido para la PCB. El segundo de ellos corresponde al editor de librerías, donde guardaremos los símbolos y huellas que hayamos ido creando a lo largo del proyecto y que no existen en las librerías del propio programa. En tercer lugar, tenemos acceso al diseño de la PCB, donde importaremos el *Netlist* del editor de esquemas para posteriormente reubicar cada uno de los componentes necesarios para el proyecto. Por último, el editor de huellas nos permitirá hacer al detalle cada una de las huellas de

los componentes del circuito electrónico.

En la realización de este proyecto se ha hecho uso de la guía de inicio rápido proporcionada por el equipo de *KiCad*, la cual puede ser consultada en la siguiente cita del apartado destinado a las *Referencias* [16].

4.7. Realterm

El *Realterm* [17] fue utilizado durante la realización del proyecto principalmente en la fase previa al diseño de la interfaz gráfica, cuyo principal objetivo fue comprobar la máquina de estados integrada en la FPGA para el envío de parámetros.

Para ello, los únicos cambios que se realizaron nada más arrancar este programa son los vistos en la figura 4.29, donde se especifica el puerto serie al que nos conectamos, los baudios a los que se realiza la comunicación, así como el número de bits de datos, paridad, bits de parada, entre otros aspectos.

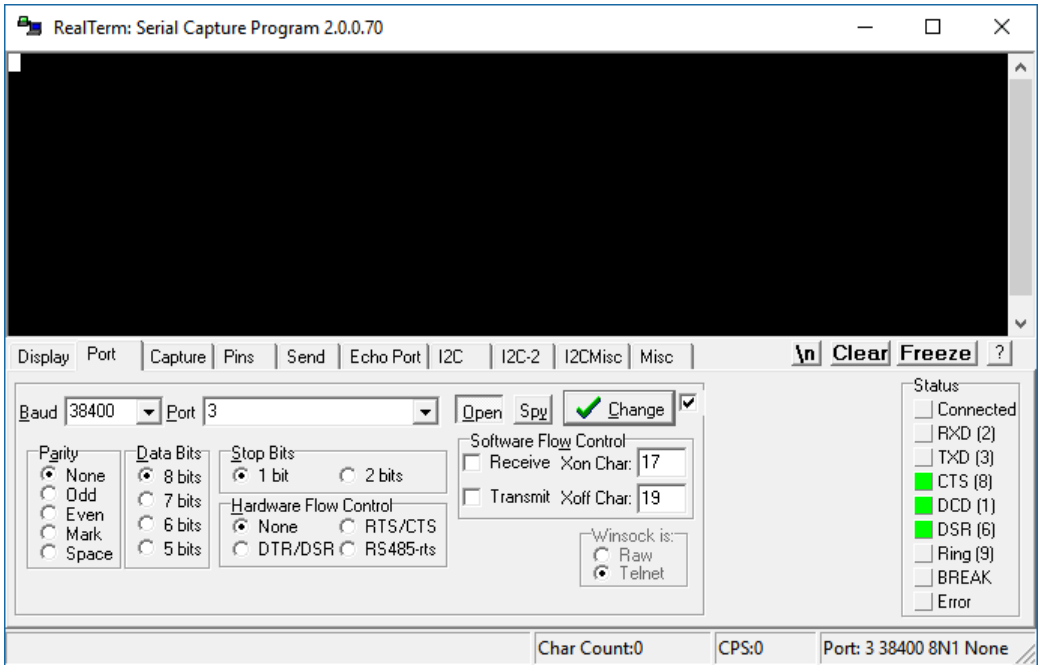


Figura 4.29: Software para la comprobación de la comunicación serie *Realterm*. *Realterm*.

Aparte de los cambios ya comentados, en la pestaña *Display* podemos cambiar el formato en pantalla de los datos recibidos. Por último, en la pestaña *Send*, enviare-

mos el número que corresponda con el estado de la máquina de estados que queramos comprobar.

Parte III

Descripción del Proyecto

Capítulo 5

Hardware

5.1. Estructura General

La parte hardware del proyecto realizada sobre FPGA se divide en diferentes módulos que son unidos en el fichero denominado *SGandO.vhd* y cuyo código puede verse en el anexo destinado a los mismos. La figura 5.1 muestra la estructura general del proyecto realizado. Los componentes hardware que dan forma al sistema más abstracto del proyecto son:

- **signal_generator:** Incluye todos los módulos que son específicos del generador de señales.
- **oscilloscope:** Incluye todos los módulos que son específicos del osciloscopio.
- **state_machine:** Máquina de estados para la comunicación entre PC y FPGA.
- **UART:** Protocolo de comunicación serie.
- **rst_module:** Módulo creado únicamente para el reset general de todo el proyecto nada más configurar la FPGA.

Aparte de estos componentes, y como ya veremos en próximos apartados, dos paquetes contienen las constantes y funciones utilizadas en todo el proyecto hardware, denominadas *parametros.vhd* y *real2bit.vhd*.

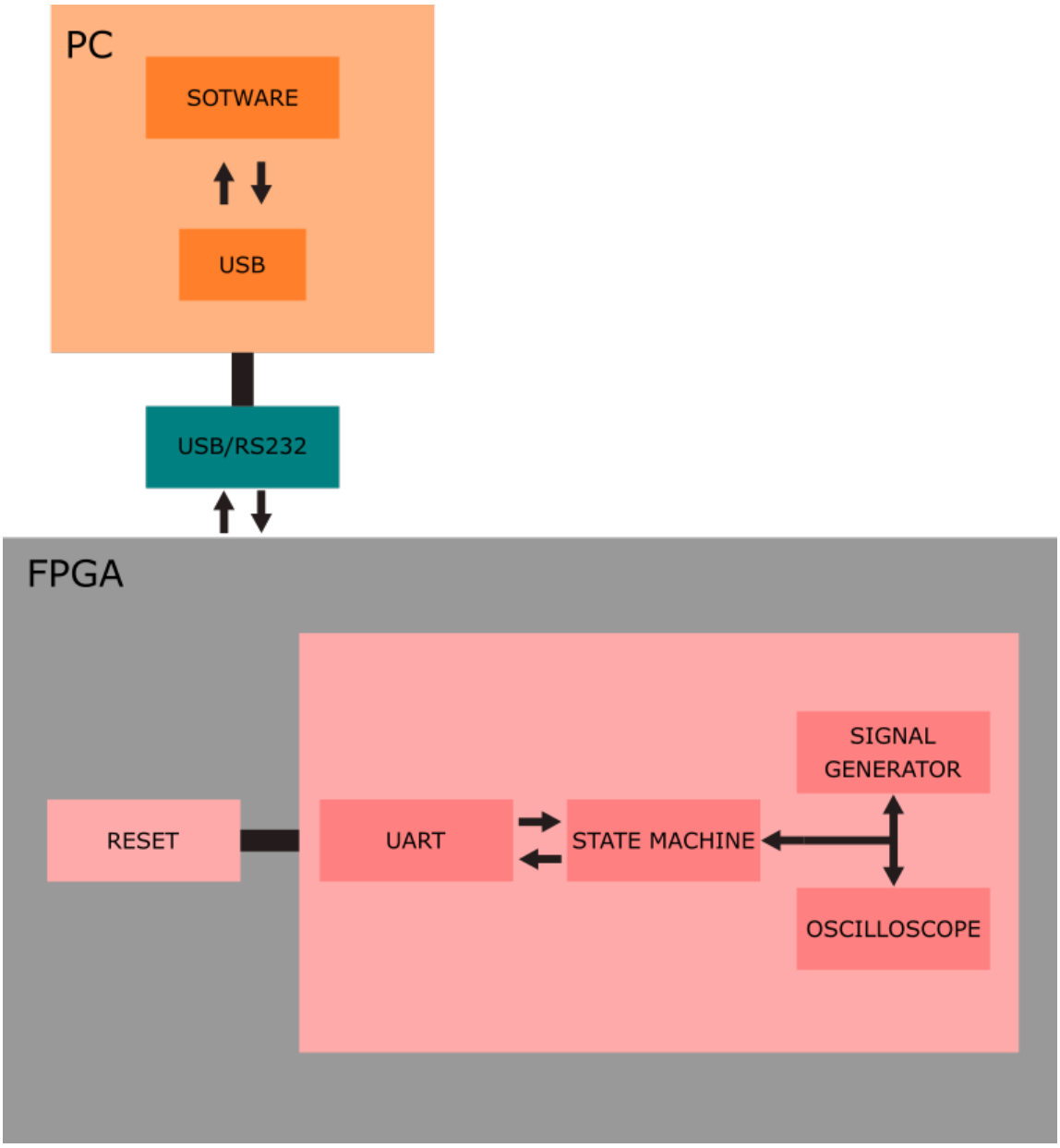


Figura 5.1: Estructura general de *SGandO.vhd*. *Elaboración propia.*

Las entradas a *SGandO.vhd* y, por lo tanto, a la FPGA configurada son:

- **data_ADC(11:0):** Datos de entrada (12 bits) del ADC de la placa realizada.
- **sel_signal(1:0):** Selectores $< SW0 y SW1 >$ para generar las señal cuando el control del generador de señales lo tiene la placa de *Xilinx*.

  ”00”: Señal senoidal.

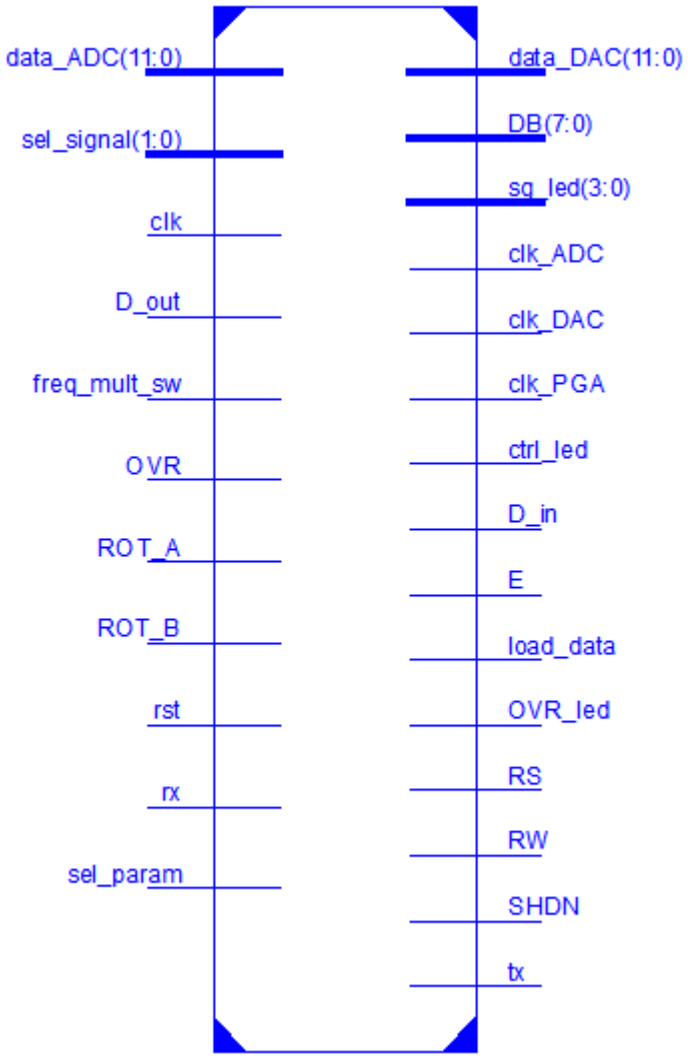


Figura 5.2: I/O de *SGandO.vhd*. *Elaboración propia.*

"01": Señal triangular.

"11": Señal cuadrada.

"10": Señal continua.

- **clk:** Entrada de reloj de la placa de 50 MHz.
- **D_out:** Línea de confirmación de datos del PGA de la placa realizada.
- **freq_mult_sw:** Selector $< SW3 >$ para el multiplicador de frecuencia cuando el control del generador de señales lo tiene la placa de *Xilinx*.

- **OVR:** Línea de señal saturada proveniente del ADC.
- **ROT_A y ROT_B:** Dirección a la que rota el interruptor rotativo.
- **rst:** Reset general del sistema en el pulsador *SOUTH* de la placa.
- **rx:** Entrada de datos del puerto serie.
- **sel_param:** Pulsador del interruptor rotativo para la selección de parámetros cuando el control del generador de señales lo tiene la placa de *Xilinx*.

Las salidas de *SGandO.vhd* y, por lo tanto, de la FPGA configurada son:

- **data_DAC(11:0):** Datos de salida (12 bits) al DAC de la placa realizada.
- **DB(7:0):** Datos de entrada y salida del display LCD para la visualización de los distintos parámetros que controlan el generador de señales cuando el control del mismo lo tiene la placa de *Xilinx*.
- **sg_led(3:0):** Indicadores LED para conocer la señal que se está generando:

  $LED < 3 >$: Señal senoidal.

  $LED < 2 >$: Señal triangular.

  $LED < 1 >$: Señal cuadrada.

  $LED < 0 >$: Señal continua.

- **clk_ADC:** Reloj del ADC de la placa realizada.
- **clk_DAC:** Reloj del DAC de la placa realizada.
- **clk_PGA:** Reloj del PGA de la placa realizada.
- **ctrl_led:** Indicador LED ($LED < 7 >$) para saber quién controla el generador de señales:

  '0': El generador de señales es controlado desde el software.

  '1': El generador de señales es controlado desde la placa

- **D_in:** Datos de salida para nueva ganancia del PGA de la placa realizada.
- **E:** Señal de *enable* del display LCD.
- **load_data:** Señal para habilitar la carga de datos en el PGA de la placa realizada.
- **OVR_led:** Indicador LED ($LED < 6 >$) que señala cuando existe saturación a la salida del ADC.
- **RS:** Selector de registros del display LCD.
- **RW:** Control de lectura y escritura del display LCD.
- **SHDN:** Señal de reset del PGA.
- **tx:** Salida de datos hacia el puerto serie.

5.2. Paquetes

El conjunto de constantes que existen en todos los códigos VHDL del proyecto son definidas en el fichero *parametros.vhd*.

Estos dos códigos se encuentran en el anexo destinado a los códigos realizados.

5.3. Funciones

El fichero *real2bit.vhd*, cuyo creador fue uno de los tutores de este proyecto, *Oswaldo González*, ha sido modificado en parte para llevar a cabo las acciones necesarias para la generación de diferentes señales como son la senoidal, triangular o cuadrada, todas ellas contenidas en unas tablas de longitud $M = 2^{12}$.

A parte de estas tablas de datos, se definen dos funciones que realizan la adaptación de los mismos para su salida hacia el DAC. Estas funciones son:

- **Truncar:** Divide el dato en parte entera y decimal y le asigna el signo correspondiente.

- **Extraer:** Prepara el dato para el envío con el signo que le pertenece.

El código denominado *real2bit.vhd* se encuentra en el anexo destinado a los códigos realizados.

5.4. Reset Inicial

Para la inicialización de todos los módulos que componen la parte hardware del proyecto, es necesaria la acción de un reset general. Si bien esta es posible mediante un pulsador, *SOUTH(T15)* de la placa, a su vez se realiza uno de manera automática al cargar el programa en la FPGA. Con esto evitamos tener que acordarnos de presionar el reset después de la configuración del dispositivo.

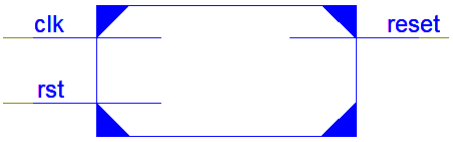


Figura 5.3: I/O del módulo *rst_module.vhd*. *Elaboración propia.*

Las entradas a este módulo son:

- **clk:** Entrada de reloj de la placa de 50 MHz.
- **rst:** Pulsador *SOUTH(T15)* de la placa.

Las salidas de este módulo son:

- **reset:** Reset general al resto de módulos del sistema.

En la figura 5.4 se muestra el funcionamiento descrito anteriormente, siendo $D8$ el reset con retardo y $D9$ el reset del pulsador de la placa.

El código denominado *rst_module.vhd* se encuentra en el anexo destinado a los códigos realizados.

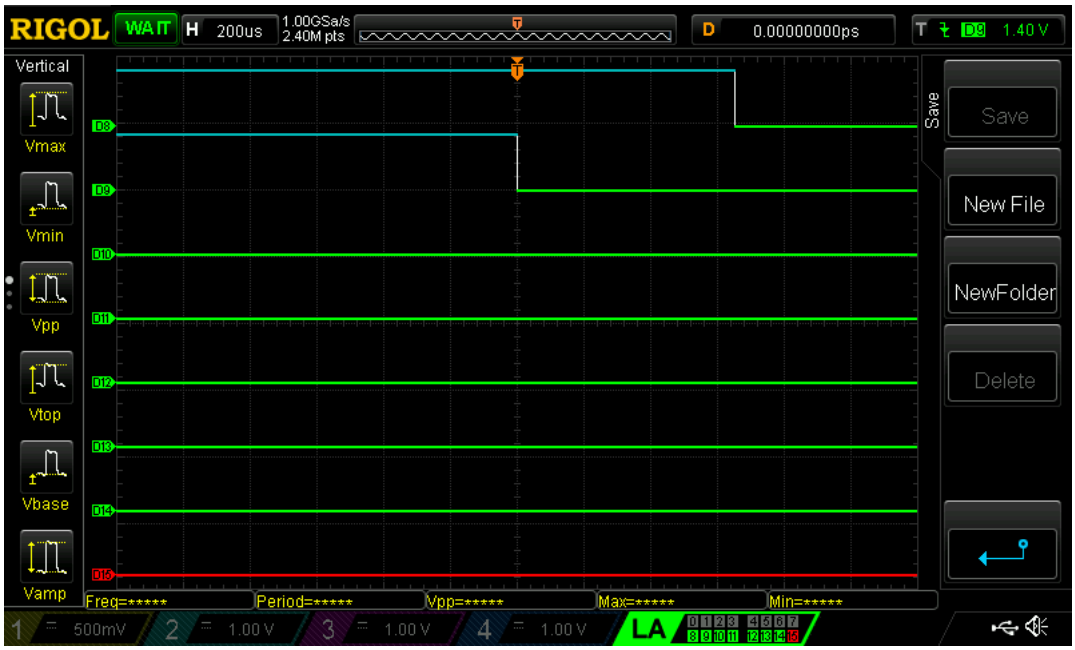


Figura 5.4: Retardo del reset general. *Imagen extraída de Rigol Oscilloscope.*

5.5. Comunicación Serie (RS–232)

El módulo UART es obra de uno de los tutores de este proyecto, *Oswaldo González*, y se trata del protocolo de comunicación serie para el envío y recepción de datos a través del puerto RS–232.

Este módulo es el encargado de comunicar la placa con el PC. Su velocidad de transmisión y recepción, medida en baudios (nº bits/s), será de 38400 y no se incluirá el bit de paridad en la comunicación. El módulo actuará conjuntamente con la máquina de estados del sistema; cuando el PC envía la señal por 'rx', ésta es almacenada en el *buffer* de entrada de la UART y, después de tratar dicha señal, convirtiéndola en un vector, es enviada a la máquina de estados. De manera inversa, todos los datos que el ADC de la placa realizada vaya tomando, serán enviados de nuevo a la UART en forma de vector y ésta la convertirá en señal serie para su envío por medio de 'tx' hacia el PC.

Las entradas a la UART son:

- **data_in(7:0):** Byte de datos a transmitir.

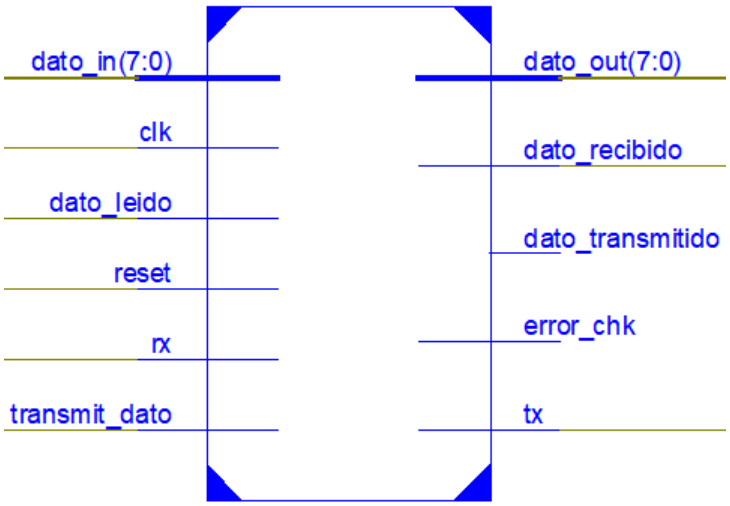


Figura 5.5: Protocolo comunicación serie. *Elaboración propia.*

- **clk:** Reloj de 50 MHz de la placa.
- **dato_leido:** Línea indicada por la máquina de estados de que el dato enviado por el módulo ha sido transmitido.
- **reset:** Señal para el reset del módulo.
- **rx:** Señal de recepción serie.
- **transmit_dato:** Línea que indica un nuevo dato para transmitir.

Las salidas de la UART son:

- **dato_out(7:0):** Byte de datos que hemos recibido desde el PC.
- **dato_recibido:** Línea que indicará a la máquina de estados cuándo se ha recibido un dato por parte del PC. Se activa a '1' cuando hay un nuevo dato entrante en la UART.
- **dato_transmitido:** Línea que indicará cuándo la UART ha transmitido el dato que se le ha suministrado.
- **error_chk:** Bit indicativo de error. Para nuestro proyecto no tendrá finalidad alguna.

- **tx:** Señal de transmisión serie.

5.6. Máquina de estados

Por medio de la máquina de estados, el software de PC es capaz de comunicarse con la FPGA para el envío de los diferentes parámetros que harán posible tanto la generación como la visualización de diferentes formas de onda.

En primer lugar, se realiza un divisor del reloj de la placa para la actualización de los datos tanto en el proceso denominado *maq_estados* como en el denominado *osc_unit*, este último, encargado de la obtención de datos proporcionados por el ADC de la placa realizada. Este reloj, cuyo funcionamiento puede verse en la figura 5.6, trabaja a una frecuencia $f = 1{,}5KHz$ aproximadamente, es decir, el periodo de actualización de los datos de entrada y salida del puerto de comunicación serie se realiza cada $0{,}5ms$ aproximadamente.

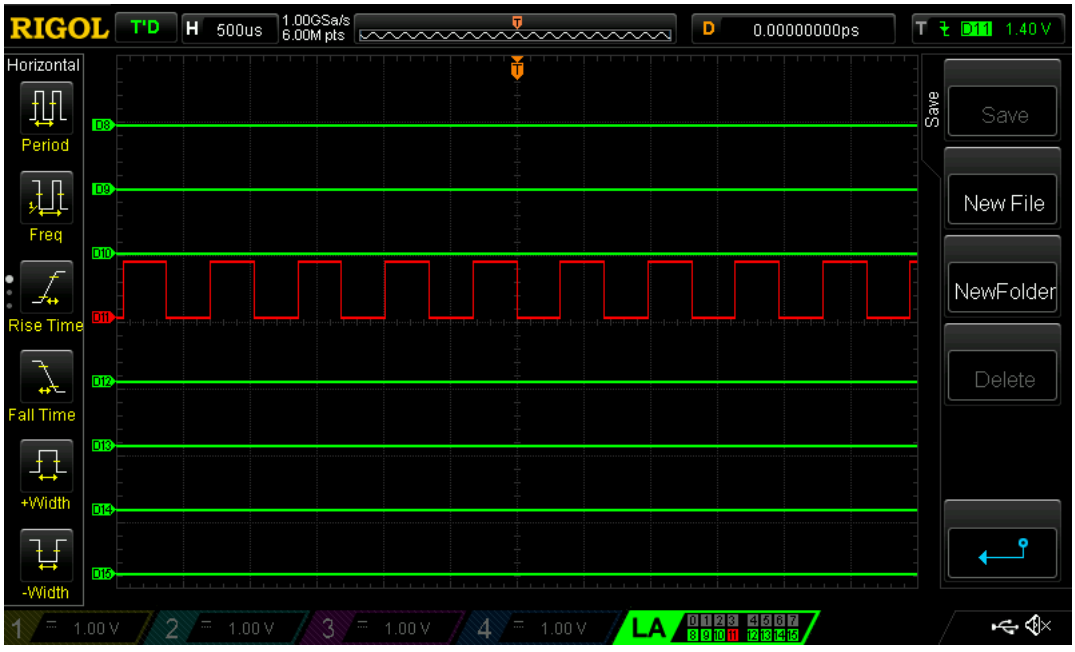


Figura 5.6: Reloj de comunicación de la máquina de estados con la UART. *Imagen extraída de Rigol Oscilloscope.*

El proceso *maq_estados* recibe el byte de datos procedente del puerto RS–232 por la

entrada *data_in*. Dependiendo del dato, la FPGA responderá de una manera u otra. A continuación, se detalla la respuesta de la máquina de estados según el dato recibido por *data_in*:

- **"00000000":** Un byte de ceros pondrá al generador de señales fuera de servicio, es decir, enviando a la salida del DAC de la placa realizada únicamente ceros. Esto significa, que el generador no responderá a estímulos ni de la placa ni del software, hasta que se vuelva a establecer la conexión del mismo.
- **"00011111":** Siendo '0' los tres bits más significativos y el resto puestos a '1', la máquina de estados ordena lo contrario que la instrucción anterior, es decir, pondrá en servicio el generador de señales.
- **"00000011":** Siendo los dos bits menos significativos un '1' lógico y el resto '0', el generador de señales responderá ante estímulos recibidos desde el software de PC.
- **"00000010":** En caso de que el mensaje de entrada sea un 2 natural en número binario de 8 bits, el generador de señales estará controlado desde la placa, ignorando la entrada de parámetros que pueda entrar por el software de PC.
- **"00000001":** Si el único bit a '1' de este vector de datos de entrada es el menos significativo, se llevará a cabo el reset de la máquina de estados, inicializando todos los parámetros a sus valores por defecto.
- **"00001010":** El número 10 en binario enviará al bloque del generador de señales la orden de que la forma de onda a generar será del tipo senoidal.
- **"00001010":** El número 10 en binario enviará al bloque del generador de señales la orden de que la forma de onda a generar será del tipo senoidal.
- **"00001011":** El número 11 en binario enviará al bloque del generador de señales la orden de que la forma de onda a generar será del tipo triangular.
- **"00001100":** El número 12 en binario enviará al bloque del generador de señales la orden de que la forma de onda a generar será del tipo cuadrada.

- **"00001101":** El número 13 en binario enviará al bloque del generador de señales la orden de que la forma de onda a generar será del tipo continua. Este último caso, aunque se encuentre disponible en la máquina de estados, el software de PC no posee acción alguna que la active, únicamente será posible generarla cuando el control del generador de señales lo tenga la placa, que lo realizará mediante los selectores de la misma.

- **"00000100":** Si lo que entra es el número 4 en binario, la frecuencia de la señal que es generada no se verá modificada.

- **"00000101":** Sin embargo, si tenemos el mismo byte anterior pero con el bit menos significativo a '1', la frecuencia de la señal generada se multiplicará por 100 antes de su envío al DAC de la placa realizada.

- **"00000110":** Para poder generar una señal con una frecuencia de $1MHz$, el reloj del DAC de la placa realizada debe ser de una frecuencia diferente que para generar el resto de frecuencias.

La modificación de los parámetros para llevar a cabo la generación y visualización de señales por medio del software de PC se lleva a cabo dependiendo de los tres bits más significativos del dato de entrada por *data_in*:

- **"001":** Cambio de la frecuencia de la señal generada. Los cinco bits menos significativos corresponderán a la frecuencia deseada.

- **"010":** Cambio en la amplitud de la señal generada. Los cinco bits menos significativos corresponderán al voltaje pico a pico de la señal generada.

- **"011":** Cambio de offset de la señal generada. Los cinco bits menos significativos corresponderán al offset añadido a la señal generada.

- **"100":** Cambio del ciclo de trabajo de la señal generada. Los cinco bits menos significativos corresponderán al ciclo de trabajo de la señal generada.

- **"101":** Cambio de la frecuencia del reloj del ADC de la placa realizada. Los cinco bits menos significativos corresponderán al divisor de la señal de reloj deseada para la adquisición de datos desde el ADC de la placa realizada.
- **"110":** Cambio en el umbral de disparo para la captura de datos del ADC. Los cinco bits menos significativos corresponderán al nivel del umbral de disparo mencionado.
- **"111":** Cambio en la ganancia del amplificador programable o PGA de la placa realizada. Los cinco bits menos significativos corresponderán a la ganancia que queremos programar en el PGA.

El otro proceso dentro de *state_machine* es *osc_unit*, que se encargará de adquirir los datos recogidos por el bloque del osciloscopio y enviarlos hacia el módulo de la UART para su envío posterior al display de la aplicación software de *SGandO*.

Las entradas al módulo de *state_machine* son:

- **data_in(7:0)** Byte de datos procedentes de la UART con la orden del software de PC.
- **osc_data(7:0):** Datos provenientes del ADC instalado en la placa realizada.
- **sel_signal_in(1:0):** Vector de señal a generar conectada a dos selectores ubicados en la placa. Únicamente tendrán efecto cuando el generador de señales sea controlado desde la placa.
- **clk:** Reloj de 50 MHz de la placa.
- **dato_recibido:** Línea que indicará a la máquina de estados cuándo se ha recibido un dato por parte del PC. Se activa a '1' cuando hay un nuevo dato entrante en la UART.
- **reset:** Señal para el reset del módulo.

Por el otro lado, las salidas del módulo *state_machine* son:

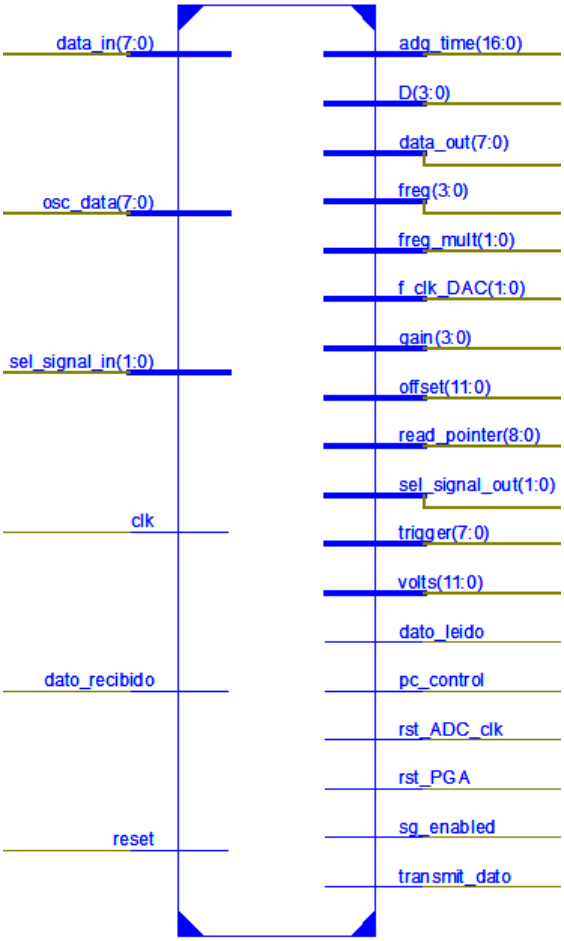


Figura 5.7: Máquina de estados de *SGandO*. *Elaboración propia*.

- **adq_time(16:0):** Divisores de tiempo del reloj del ADC instalado en la placa realizada. Estos valores pueden ser consultados en el código en VHDL *state_machine.vhd* que se encuentra en el anexo destinados a los mismos.
- **D(3:0):** Vector para describir el ciclo de trabajo de la señal generada.
- **data_out(7:0):** Salida de los datos que entran por *osc_data* a la frecuencia del reloj mencionado en este mismo apartado (figura 5.6).
- **freq(3:0):** Vector para determinar la frecuencia de la señal a generar.
- **freq_mult(1:0):** Multiplicador de frecuencia de la señal a generar.
- **f_clk_DAC(1:0):** Divisor del reloj del DAC instalado en la placa realizada.

- **gain(3:0):** Vector para indicar la ganancia del PGA instalado en la placa realizada.
- **offset(11:0):** Vector de 12 bits que forma una 'palabra' para indicar el offset que se le añade a la señal generada por *signal_generator*.
- **read_pointer(8:0):** Puntero de lectura para indicarle al módulo de osciloscopio el dato que le tiene que enviar a la máquina de estados.
- **sel_signal_out(1:0):** Le indica al módulo *signal_generator* la forma de la señal que se tiene que generar.
- **trigger(7:0):** Nivel del umbral de disparo del osciloscopio.
- **volts(11:0):** Voltaje pico a pico de la señal que se generará.
- **dato_leido:** Línea que indica de que el dato enviado por el módulo UART ha sido transmitido.
- **pc_control:** Señal que determina si el control del generador de señales lo tiene la placa o el software de PC.
- **rst_ADC_clk:** Reset del reloj del ADC instalado en la placa realizada.
- **rst_PGA:** Reset del PGA instalado en la placa realizada.
- **sg_enabled:** Habilitar/Deshabilitar el generador de señales.
- **transmit_dato:** Línea que indica a la UART un nuevo dato para transmitir.

El código denominado *state_machine.vhd* se encuentra en el anexo destinado a los códigos realizados.

5.7. Generador de Señales

El módulo de *signal_generator* será el encargado de generar las señales con los parámetros dados por el software de PC o a través de los actuadores de la placa, según la

configuración del mismo. Dentro de este bloque, tres módulos se integran en su funcionamiento:

- **DAC:** Controla el funcionamiento del DAC ubicado en la placa realizada.
- **param_SG**: Cuando el generador de señales es controlado desde la placa, será el encargado de traducir las órdenes que se le envíe al generador.
- **LCD:** Controlador del display LCD de la placa.

Además de los componentes anteriormente citados, el proceso denominado *GEN_SIGNAL* se encargará del recorrido de las tablas generadas en el paquete *real2bit* según los parámetros de frecuencia establecidos por el usuario.

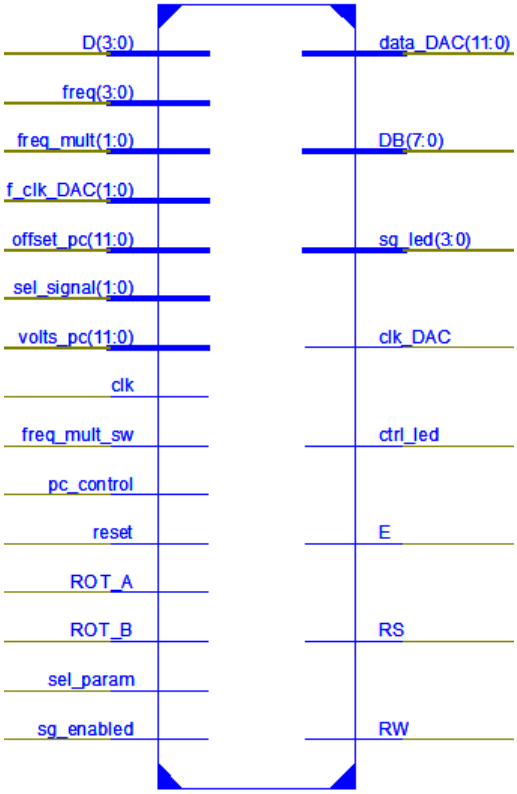


Figura 5.8: Módulo del generador de señales. *Elaboración propia.*

Las entradas de este módulo son:

- **D(3:0):** Vector para describir el ciclo de trabajo de la señal generada.

- **freq(3:0):** Vector para determinar la frecuencia de la señal a generar.
- **freq_mult(1:0):** Multiplicador de frecuencia de la señal a generar.
- **f_clk_DAC(1:0):** Divisor del reloj del DAC instalado en la placa realizada.
- **offset_pc(11:0):** Vector de 12 bits que forma una 'palabra' para indicar el offset que se le añade a la señal generada por *signal_generator*.
- **sel_signal(1:0):** Le indica al módulo *signal_generator* la forma de la señal que se tiene que generar.
- **volts_pc(11:0):** Voltaje pico a pico de la señal que se generará.
- **clk:** Reloj de 50 MHz de la placa.
- **freq_mult_sw:** Señal que proviene del selector $SW < 3 >$ de la placa y que realiza la función de multiplicador de frecuencia cuando la placa tiene el control sobre el generador de señales.
- **pc_control:** Señal que determina si el control del generador de señales lo tiene la placa o el software de PC.
- **reset:** Señal para el reset del módulo.
- **ROT_A y ROT_B:** Dirección a la que rota el interruptor rotativo.
- **sel_param:** Pulsador del interruptor rotativo para la selección de parámetros cuando el control del generador de señales lo tiene la placa.
- **sg_enabled:** Habilitar/Deshabilitar el generador de señales.

Por otro lado, las salidas del módulo *generator_signal* son:

- **data_DAC(11:0):** Datos de salida (12 bits) al DAC de la placa realizada.

- **DB(7:0):** Datos de entrada y salida del display LCD para la visualización de los distintos parámetros que controlan el generador de señales cuando el control del mismo lo tiene la placa de *Xilinx*.
- **sg_led(3:0):** Indicadores LED para conocer la señal que se está generando:

  $LED < 3 >$: Señal senoidal.

  $LED < 2 >$: Señal triangular.

  $LED < 1 >$: Señal cuadrada.

  $LED < 0 >$: Señal continua.
- **clk_DAC:** Reloj del DAC de la placa realizada.
- **ctrl_led:** Indicador LED ($LED < 7 >$) para saber quién controla el generador de señales:

  '0': El generador de señales es controlado desde el software.

  '1': El generador de señales es controlado desde la placa
- **E:** Señal de *enable* del display LCD.
- **RS:** Selector de registros del display LCD.
- **RW:** Control de lectura y escritura del display LCD.

5.7.1. Conversión Digital → Analógico (DAC)

La conversión de los datos que se proporcionan en un vector de 12 bits a la placa que se realizó por necesidad del proyecto, se realiza en el convertidor digital → analógico *AD9752* [18]. *DAC.vhd* es el módulo encargado de proporcionar este vector, el cual posee tres procesos.

El primero de ellos, **CLK_ENVIOS_DAC_A** (figura 5.9), le proporciona al DAC una frecuencia de captura de 400 KHz cuya misión es la generación de señales de **frecuencias comprendidas entre 150 Hz y 10 KHz**, estando activo el multiplicador *x1* del generador de señales.

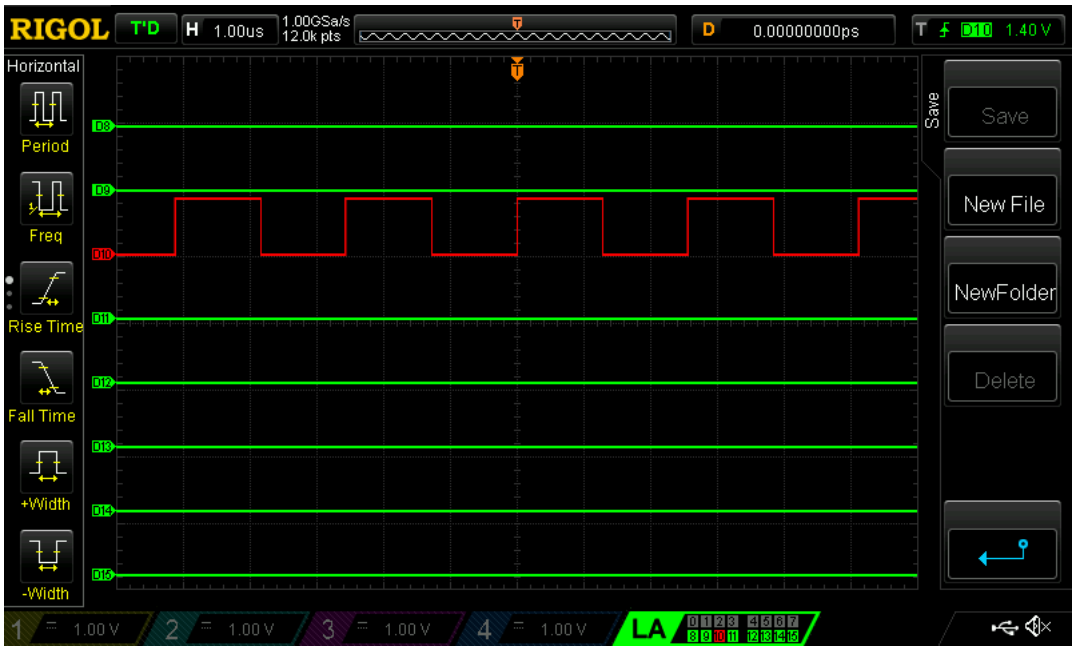


Figura 5.9: Reloj para generación de frecuencias en KHz. *Imagen extraída de Rigol Oscilloscope.*

CLK_ENVIOS_DAC_B (figura 5.10), genera un reloj con una frecuencia de 25 MHz necesaria para la **generación de señales de 1 MHz**. Este reloj únicamente se utiliza cuando el multiplicador de frecuencia se encuentra en el estado *x100* y el dial de frecuencia en *10.0*.

Por último, la generación de **frecuencias que van desde los 150 KHz hasta los 5 MHz** se realiza a una frecuencia de reloj igual al de la placa, es decir, a 50 MHz (figura 5.11).

Las entradas y salidas del módulo del DAC se pueden ver en el siguiente esquema:

Donde las entradas son:

- **data_in(11:0):** Datos de entrada (12 bits) previos al DAC.
- **f_clk_DAC(1:0):** Divisor del reloj del DAC instalado en la placa realizada.
- **clk:** Reloj de 50 MHz de la placa.
- **reset:** Señal para el reset del módulo.

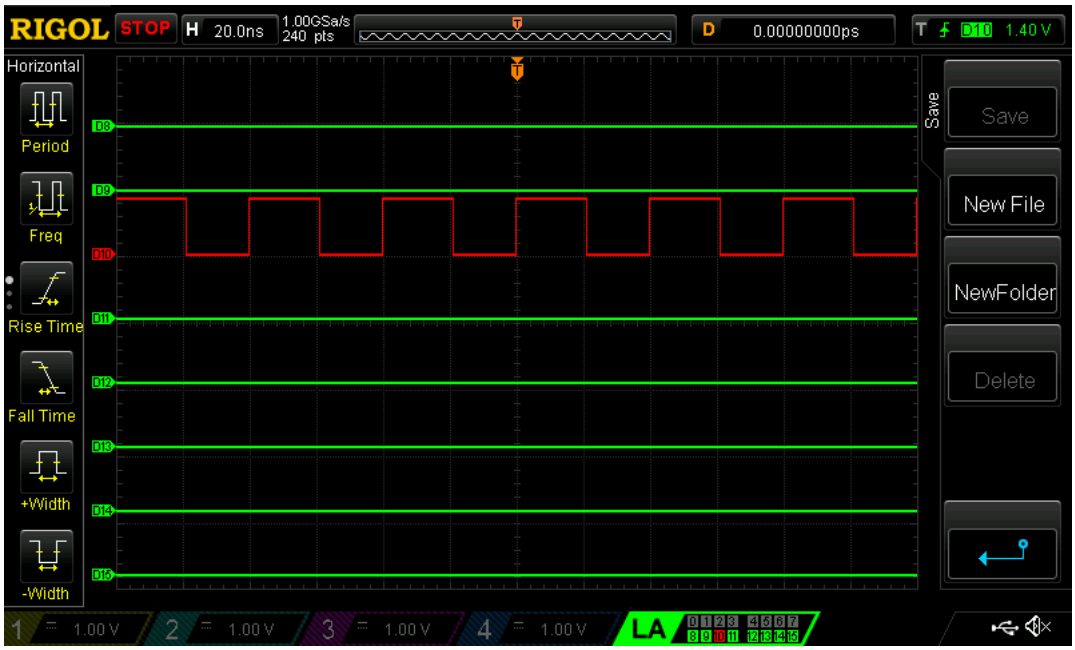


Figura 5.10: Reloj para generación de frecuencias igual a 1 MHz. *Imagen extraída de Rigol Oscilloscope.*

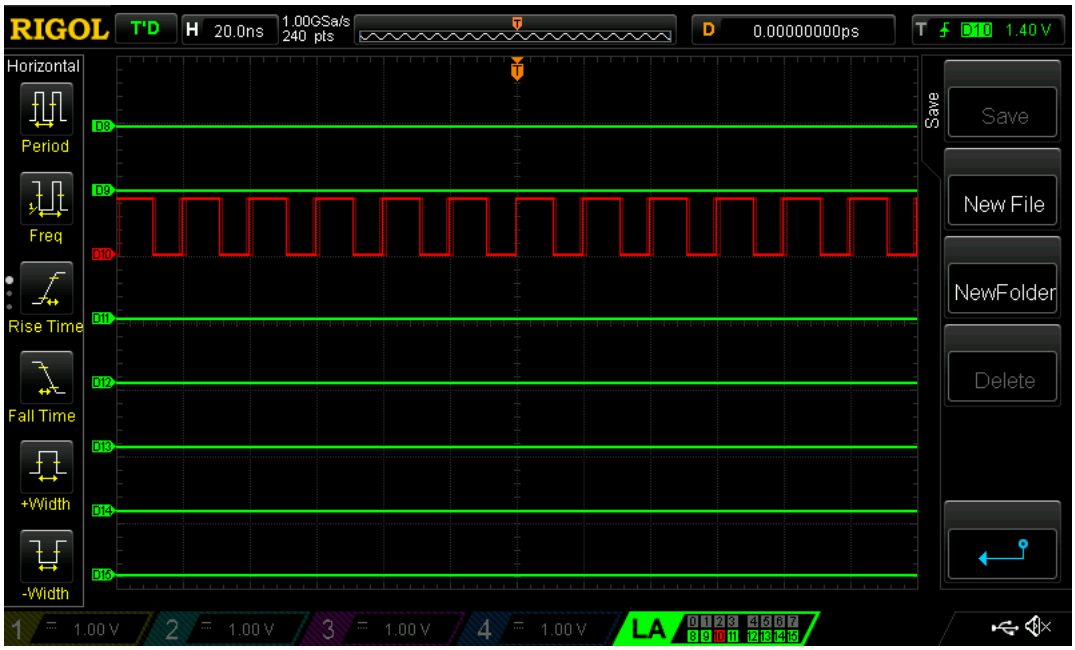


Figura 5.11: Reloj para generación de frecuencias en MHz. *Imagen extraída de Rigol Oscilloscope.*

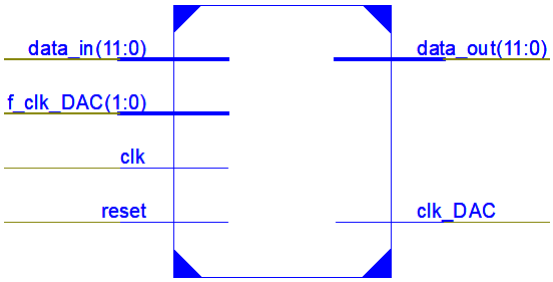


Figura 5.12: Módulo DAC del bloque de generador de señales. *Elaboración propia.*

Las salidas de este módulo son:

- **data_out(11:0):** Datos de salida (12 bits) al DAC de la placa realizada.
- **clk_DAC:** Reloj del DAC de la placa realizada.

El código denominado *DAC.vhd* se encuentra en el anexo destinado a los códigos realizados.

5.7.2. Selección de Parámetros

El módulo *param_SG* es el encargado de proporcionar los parámetros necesarios para la generación de las señales, bien sea controlado desde el software de PC o bien desde la placa.

Uno de los tres procesos que conforman este módulo es **EVENTOS_ROT_A_B**, encargado del cambio de cada uno de los parámetros. Su nombre se debe a que el cambio de producirá cuando haya un evento de rotación en el interruptor rotativo de la placa cuando el control del generador de señales lo tenga la misma. Sin embargo, este mismo proceso, en caso de que la entrada *pc_control* sea '1', es decir, que el software de PC tiene el control del generador de señales, actuará como puente entre los datos que proporciona la máquina de estados con las instrucciones del software y la generación de datos en el DAC. Los parámetros que pueden ser modificados son: multiplicador de frecuencia, voltaje pico a pico de la señal, offset y ciclo de trabajo, este último, es el único de todos ellos que siguen las mismas instrucciones dentro del código para indicar

el ciclo de trabajo de la señal generada. Para poder saber en qué sentido ha girado el interruptor rotativo, se hace uso del componente *rotary_switch*.

Otro de los procesos es **CAMBIO_PARAMETRO**, el cual solo será empleado cuando el control del generador de señales lo tenga la placa. Su misión es señalarle al proceso EVENTOS_ROT_A_B el parámetro que debe modificar al actuar sobre el interruptor rotativo. Estas modificaciones se realizarán haciendo presión sobre este interruptor, cuyo estado será indicado con el símbolo '>' en el display LCD de la placa.

El proceso que queda por comentar dentro de este módulo de *param_SG* es el **SALTOS_ TABLA**, que se encargará de enviar el dato al módulo DAC para su envío al DAC físico.

```
SALTOS_TABLA : process(clk, reset)
begin
   if reset = '1' then
      freq_out <= 1;
   elsif clk'event and clk = '1' then
      if position > M/2 then
         freq_out <= escala*freq_sig*D_sig/DCycle;
      else
         freq_out <= escala*freq_sig*(DCycle-D_sig)/DCycle;
      end if;
   end if;
end process SALTOS_TABLA;
```

Según la posición (*position*) del puntero *pointer* que se encuentra dentro del módulo general del generador de señales, la salida será una o la otra. Esta salida varía entre la parte positiva de la señal en la tabla creada para ello en el fichero *real2bit.vhd* y la parte negativa, estando el paso por '0' en $M/2$, donde $M = 2^{12}$. De esta manera, desde el momento que *position* sea mayor que $M/2$, la señal cambiará de signo.

Por último, desde este mismo módulo se envían los datos al display LCD para poder visualizar los parámetros que queremos modificar de la señal que queramos generar

siempre y cuando el control del generador lo tenga la placa y no el PC.

Las entradas y salidas de *param_sg* se muestran en la figura 5.13:

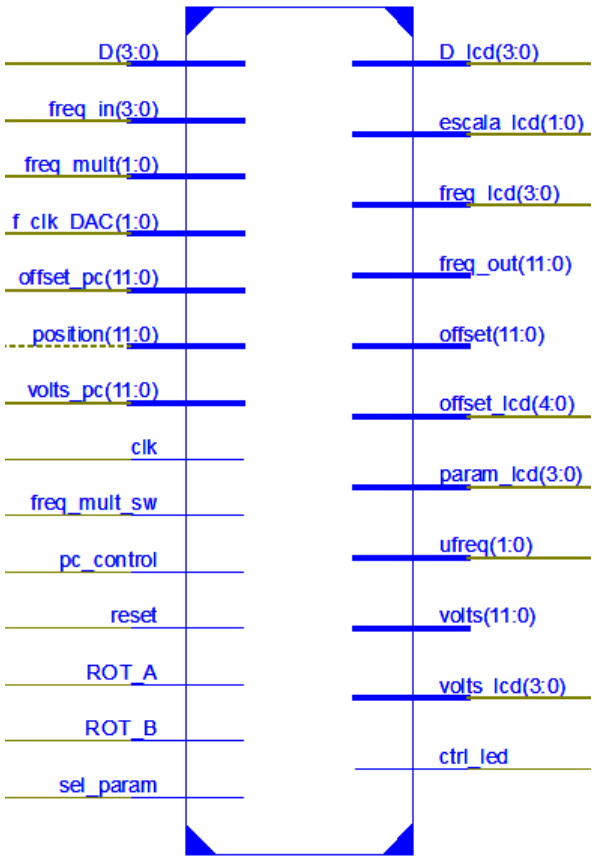


Figura 5.13: Módulo de selección y acondicionamiento de parámetros del bloque de generador de señales. *Elaboración propia*.

Donde, concretamente, las entradas son:

- **D(3:0):** Vector para describir el ciclo de trabajo desde el PC de la señal generada.
- **freq_in(3:0):** Vector para determinar desde el PC la frecuencia de la señal a generar.
- **freq_mult(1:0):** Multiplicador de frecuencia de la señal a generar.
- **f_clk_DAC(1:0):** Divisor del reloj del DAC instalado en la placa realizada.

- **offset_pc(11:0):** Vector de 12 bits que forma una 'palabra' para indicar el offset que se le añade a la señal generada por *signal_generator*.
- **position(11:0):** Vector de 12 bits que indica la posición en la que se encuentra el puntero *pointer* que apunta a las tablas creadas en el fichero *real2bit.vhd*.
- **volts_pc(11:0):** Voltaje pico a pico de la señal que se generará.
- **clk:** Reloj de 50 MHz de la placa.
- **freq_mult_sw:** Señal que proviene del selector $SW < 3 >$ de la placa y que realiza la función de multiplicador de frecuencia cuando la placa tiene el control sobre el generador de señales.
- **pc_control:** Señal que determina si el control del generador de señales lo tiene la placa o el software de PC.
- **reset:** Señal para el reset del módulo.
- **ROT_A y ROT_B:** Dirección a la que rota el interruptor rotativo.
- **sel_param:** Pulsador del interruptor rotativo para la selección de parámetros cuando el control del generador de señales lo tiene la placa.

Las salidas con las que cuenta *param_SG* son:

- **D_lcd(3:0):** Ciclo de trabajo de la señal generada que se muestra en el display LCD cuando el control de parámetros lo tiene la placa.
- **escala_lcd(1:0):** Escala de la frecuencia de la señal generada que se muestra en el display LCD cuando el control de parámetros lo tiene la placa.
- **freq_lcd(3:0):** Frecuencia de la señal generada que se muestra en el display LCD cuando el control de parámetros lo tiene la placa.
- **freq_out(11:0):** Saltos que habrán en las tablas de señales para su generación a la frecuencia especificada.

- **offset(11:0):** Indica el offset de la señal a generar.
- **offset_lcd(4:0):** Offset de la señal generada que se muestra en el display LCD cuando el control de parámetros lo tiene la placa.
- **param_lcd(3:0):** Selección del parámetro que se quiere modificar en el display LCD cuando el control de los mismos se lleva a cabo desde la placa.
- **ufreq(1:0):** Marca las diferentes frecuencias a las que el DAC muestrea.
- **volts(11:0):** Indica la tensión pico a pico de la señal a generar.
- **volts_lcd(3:0)** Voltaje pico a pico de la señal generada que se muestra en el display LCD cuando el control de parámetros lo tiene la placa.
- **ctrl_led:** Indicador led (LED¡7¿) del control de parámetros de *signal_generator*.

  '1': El control del generador de señales lo tiene el software.

  '0': El control del generador de señales lo tiene la placa.

El código denominado *param_SG.vhd* se encuentra en el anexo destinado a los códigos realizados.

5.7.3. Display LCD

El módulo destinado al control de la pantalla LCD de la placa fue realizado por *Digilent®, Inc.* [19] y modificado para la adaptación al proyecto llevado a cabo. Estas modificaciones fueron únicamente de cambio de reloj de actualización y texto mostrado en la pantalla, además de eliminar salidas que no eran necesarias para la aplicación realizada.

La visualización de datos en el display corresponde únicamente a parámetros relacionados con el generador de señales, debido a que su funcionamiento no depende del software de PC. Dicho esto, cuando el generador de señales es controlado desde el PC, el display muestra la siguiente leyenda:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	C	o	n	t	r	o	l		d	e		S	.	G	.	
2	p	o	r		s	o	f	t	w	a	r	e	!	!		

Figura 5.14: Display LCD cuando el generador de señales está controlado desde el software de PC. *Elaboración propia.*

Cuando el software es controlado desde la placa, en el display se visualizarán ahora los diferentes parámetros modificables para el generador de señales. La modificación se realizará en el parámetro que tenga delante el símbolo '>', siendo posible mover este último con el pulsador del interruptor rotativo. A continuación se muestra la pantalla donde en color rojo están los caracteres que sufren modificaciones en el cambio del parámetro.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	>	f	=	x	x	.	x	K	H	z	>	D	=	x	/	x
2	>	V	=	x	.	x	x	>	-	x	.	x	x	V		

Figura 5.15: Display LCD cuando el generador de señales está controlado desde la placa. *Elaboración propia.*

Como último aspecto a tener en cuenta dentro del módulo del display LCD es el reset que se le realiza cada 0.16 segundos mediante el componente denominado *refresh_lcd* (figura 5.16), con el objetivo de ver de manera rápida los cambios en cada uno de los parámetros modificables.

Este código (*refresh_lcd*.vhd), así como el de la máquina de estados del display LCD (*LCD.vhd*), se encuentra en el anexo destinado a los códigos realizados.

5.8. Osciloscopio

El módulo *oscilloscope* integra dos componentes necesarios para la adquisición de datos muestreados por el ADC de la placa realizada para este proyecto, cuyo modelo es el *ADS805* [20] de *Texas Instruments*.

- **adq_module:** Encargado de la adquisición de datos del ADC.

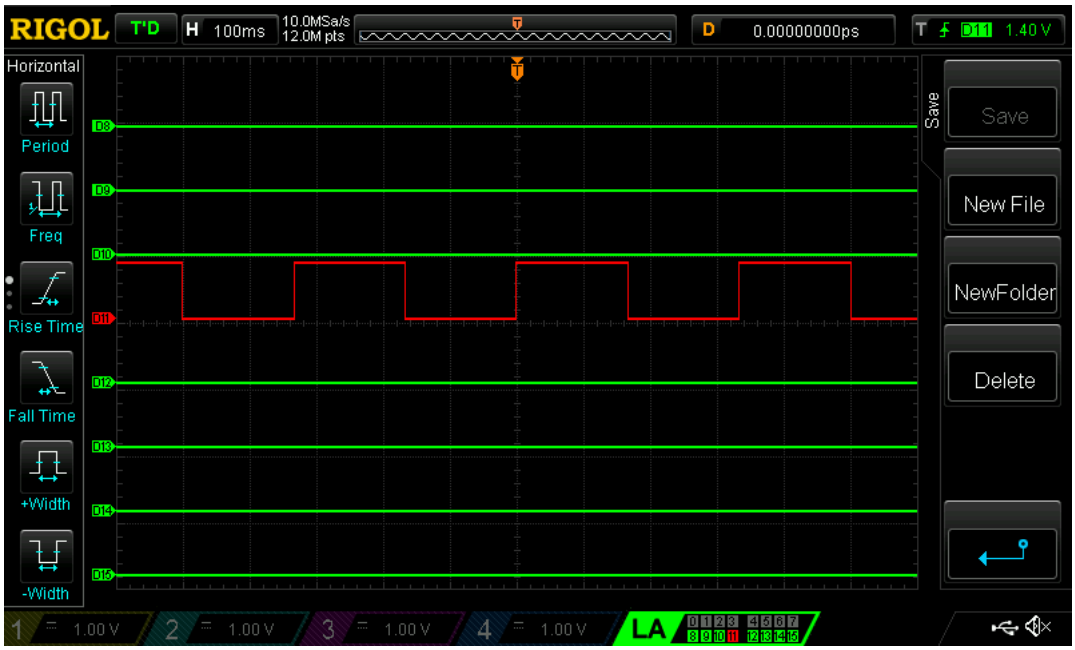


Figura 5.16: Reloj de actualización del display LCD. *Imagen extraída de Rigol Oscilloscope.*

- **PGA:** Encargado de establece la ganancia a la señal de entrada.

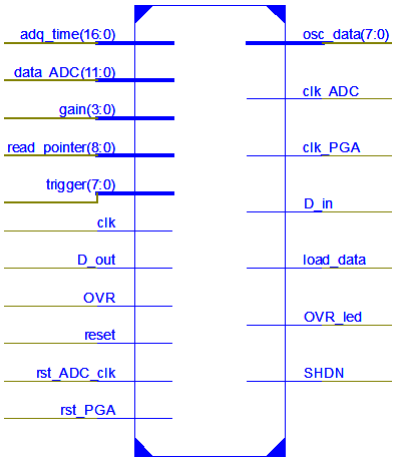


Figura 5.17: Módulo genérico del osciloscopio. *Elaboración propia.*

Las entradas al módulo general de captura de datos son las siguientes:

- **adq_time(16:0):** Divisores de tiempo del reloj del ADC instalado en la placa realizada. Estos valores pueden ser consultados en el código en VHDL *sta-*

te_machine.vhd que se encuentra en el anexo destinados a los mismos.

- **data_ADC(11:0):** Datos de entrada (12 bits) del ADC de la placa realizada.
- **gain(3:0):** Vector para indicar la ganancia del PGA instalado en la placa realizada.
- **read_pointer(8:0):** Puntero de lectura para indicarle al módulo de osciloscopio el dato que le tiene que enviar a la máquina de estados.
- **trigger(7:0):** Nivel del umbral de disparo del osciloscopio.
- **clk:** Reloj de 50 MHz de la placa.
- **D_out:** Línea de confirmación de datos del PGA de la placa realizada.
- **OVR:** Línea de señal saturada proveniente del ADC.
- **reset:** Señal para el reset del módulo.
- **rst_ADC_clk:** Reset del reloj del ADC instalado en la placa realizada.
- **rst_PGA:** Reset del PGA instalado en la placa realizada.

Así mismo, las salidas de este módulo son:

- **osc_data(7:0):** Datos provenientes del ADC instalado en la placa realizada.
- **clk_ADC:** Reloj del ADC de la placa realizada.
- **clk_PGA:** Reloj del PGA de la placa realizada.
- **D_in:** Datos de salida para nueva ganancia del PGA de la placa realizada.
- **load_data:** Señal para habilitar la carga de datos en el PGA de la placa realizada.
- **E:** Señal de *enable* del display LCD.
- **OVR_led:** Indicador LED ($LED < 6 >$) que señala cuando existe saturación a la salida del ADC.
- **SHDN:** Señal de reset del PGA.

5.8.1. Conversión Analógico → Digital (ADC)

El módulo *adq_module* tiene tres procesos que se encargan de la captura de datos del ADC que se instaló en la placa realizada. El primero de ellos, crea el reloj que se encargará de muestrear la señal de entrada, ya que para una buena visualización de los datos de entrada en una memoria de 256 datos, según la frecuencia en la entrada será el muestreo en la adquisición.

```
ADC_CLOCK : process(clk, reset, rst_adc_clk)
   variable count : integer range 0 to adq_count := 0;
   variable clk_signal : std_logic := '0';
begin
   if (reset or rst_adc_clk) = '1' then
      count := 0;
      clk_signal := '0';
      clk_ADC_sig <= '0';
   elsif clk'event and clk = '1' then
      if count = adq_time then
         clk_signal := not clk_signal;
         count := 0;
      end if;
         count := count + 1;
   end if;
clk_ADC_sig <= clk_signal;
end process;
```

En este código, la señal de entrada *adq_time* varía en función del tiempo de adquisición que requiramos. Así, por ejemplo, cuando queramos visualizar señales que tienen un periodo de 1 milisegundo o similar, $adq\_time = 2^10$. Mientras que si queremos visualizar señales de mayor frecuencia, $adq\_time < 2^4$. Este dato es suministrado por el módulo *state_machine* debido al mensaje enviado por el software de PC.

El segundo de los procesos es el **ADQ_MODULE**, y es el encargado de la captura de datos del ADC físico. El siguiente código muestra el funcionamiento del proceso:

```
if reset = '1' then
   pointer := 0;
   trigger_pointer := L_samples/2;
   trigger_value <= (others => '0');
   trigger_state := false;
   reset_memo := false;
elsif clk_ADC_sig'event and clk_ADC_sig = '1' then

   if (trigger_value /= trigger) and (trigger_state = true) then
      reset_memo := true;
      trigger_state := false;
      trigger_value <= trigger;
      pointer := 0;
   end if;
```

Desde el momento en el que el valor de *trigger* no cambie, se procederá al reset de las memorias de adquisición con la variable declarada de *reset_memo := true*. A continuación, se procederá a la ejecución de otra serie de instrucciones.

```
case reset_memo is
   when true =>
      if pointer < L_samples then
         memo_adq(pointer) <= (others => '0');
         memo_adq_buffer(pointer) <= (others => '0');
         pointer := pointer+1;
      else
         pointer := 0;
         reset_memo := false;
      end if;
```

```
11      when others =>
12          memo_adq(pointer) <= data_in(nbits-1 downto 4);
13          pointer := (pointer + 1) mod L_samples;
14          trigger_pointer := (pointer + L_samples/2) mod L_samples;
15
16          if data_in(nbits-1 downto 4) < trigger then
17              trigger_state := true;
18          end if;
19
20          if read_pointer = 0 and trigger_state = true then
21              memo_adq_buffer(trigger_pointer downto 0) <= memo_adq(
                    trigger_pointer downto 0);
22              memo_adq_buffer(L_samples-1 downto trigger_pointer+1)
                    <= memo_adq(L_samples-1 downto trigger_pointer+1);
23          end if;
24  end case;
```

En el caso que nos ocupa, que es el instante posterior al que se ha cambiado el umbral de disparo, es decir, *reset_memo := true*, las dos memorias de adquisición son rellenas de '0s' en todas sus líneas. Estas memorias de 256 datos son:

- **memo_adq:** Es la memoria que adquiere los datos directamente del ADC y los va almacenando.

- **memo_adq_buffer:** Es la copia de *memo_adq* cuando *state_machine* así lo considere.

Una vez se recorran las dos memorias y se hayan llenado de '0s', el estado de *reset_memo* pasará a ser *false*, por lo que entraría en el segundo de los casos posibles en el próximo flanco de subida de reloj.

En este segundo estado, cuando *reset_memo := false*, intervendrán dos punteros: *pointer*, que se encargará del relleno de la memoria *memo_adq* en cada flanco de subida

de reloj, y *trigger_pointer*, cuya misión es vigilar que el dato de la posición en la que se encuentra es mayor que el valor de *trigger* especificado por el usuario. Estos dos punteros se encuentran separados entre sí una cantidad *L_samples/2*, donde $L\_samples = 256$.

Cuando el dato de entrada sea mayor que el valor del *trigger*, $trigger\_state = true$, y el puntero de envío de datos hacia *state_machine* sea igual a 0, se comenzará a pasar los datos que hay en la memoria *memo_adq* a la memoria *memo_adq_buffer*. Este relleno se llevará a cabo desde la posición en la que se encuentre *trigger_pointer* hacia atrás.

Finalmente, con la línea de código mostrada a continuación, los datos son leídos por la máquina de estados de *state_machine* para su envío al software de PC pasando antes por la UART. Este envío se irá realizando según el avance de *read_pointer*, entrada del módulo proveniente de la máquina de estados antes mencionada.

```
1 data_out <= memo_adq_buffer(read_pointer) + 2**7;
```

Las entradas y salidas a este módulo son:

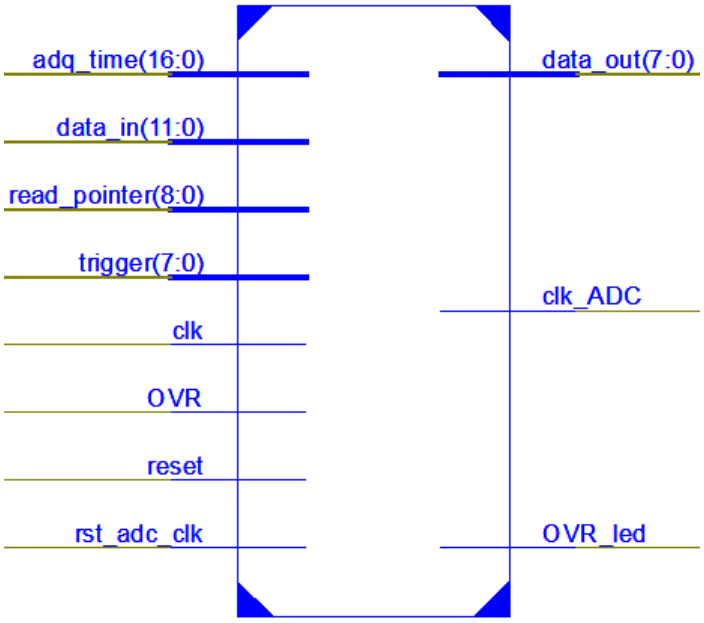


Figura 5.18: Módulo de adquisición de datos. *Elaboración propia.*

Las entradas al módulo de captura de datos son:

- **adq_time(16:0):** Divisores de tiempo del reloj del ADC instalado en la pla-

ca realizada. Estos valores pueden ser consultados en el código en VHDL *state_machine.vhd* que se encuentra en el anexo destinados a los mismos.

- **data_in(11:0):** Datos de entrada (12 bits) del ADC de la placa realizada.
- **read_pointer(8:0):** Puntero de lectura para indicarle al módulo el dato que le tiene que enviar a la máquina de estados.
- **trigger(7:0):** Nivel del umbral de disparo del osciloscopio.
- **clk:** Reloj de 50 MHz de la placa.
- **OVR:** Línea de señal saturada proveniente del ADC de la placa realizada.
- **reset:** Señal para el reset del módulo.
- **rst_ADC_clk:** Reset del reloj del ADC instalado en la placa realizada.

Así mismo, las salidas de este módulo son:

- **data_out(7:0):** Datos provenientes del ADC instalado en la placa realizada.
- **clk_ADC:** Reloj del ADC de la placa realizada.
- **OVR_led:** Indicador LED ($LED < 6 >$) que señala cuando existe saturación a la salida del ADC.

El código completo de *adq_module.vhd* se encuentra en el anexo destinado a los mismos.

5.8.2. Amplificador de Ganancia Programable (PGA)

PGA.vhd contiene dos procesos, de los cuales uno de ellos es para la creación del reloj del segundo proceso que será el mismo que para el PGA instalado en la placa realizada para este proyecto. Este reloj tendrá una frecuencia $f = 3{,}125MHz$.

El segundo proceso, denominado **PGA_CONFIG**, se trata de una máquina de estados para llevar a cabo la correcta carga de una nueva ganancia en el PGA. En la figura

que se muestra a continuación, se ha introducido en primer lugar una ganancia igual a 1 enviando el mensaje '00010001' y posteriormente se ha cambiado a una ganancia igual a 2 enviando el mensaje '00100010', donde los cuatro bits menos significativos corresponden a la ganancia del *CANAL A* y los cuatro bits más significativos al *CANAL B*.

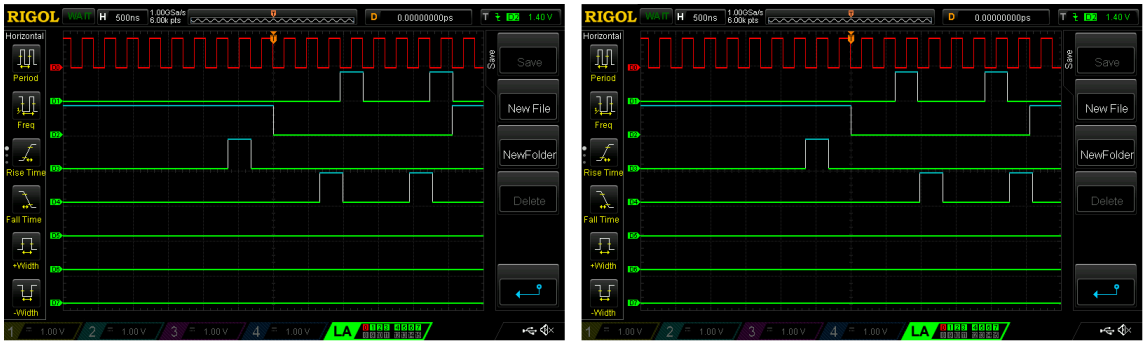


Figura 5.19: Ganancias 1 y 2 para el PGA. *Imagen extraída de Rigol Oscilloscope.*

Para conocer los datos a enviar cuando se cambia de ganancia, se adjunta el *datasheet* del PGA en la bibliografía de la memoria actual [21].

A continuación, se muestra el esquema de entradas y salidas del módulo *PGA*:

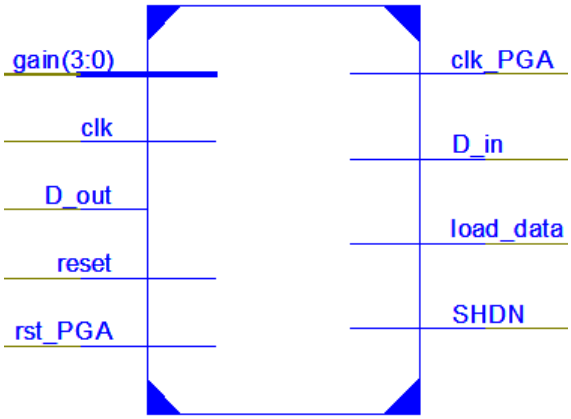


Figura 5.20: Módulo del PGA. *Elaboración propia.*

Las entradas a este módulo son:

- **gain(3:0):** Vector para indicar la ganancia del PGA.

- **clk:** Reloj de 50 MHz de la placa.
- **D_out:** Línea de confirmación de datos del PGA de la placa.
- **reset:** Señal para el reset del módulo.
- **rst_PGA:** Reset del PGA instalado.

Así mismo, las salidas de este módulo son:

- **clk_PGA:** Reloj del PGA de la placa realizada.
- **D_in:** Datos de salida para nueva ganancia del PGA.
- **load_data:** Señal para habilitar la carga de datos en el PGA.
- **SHDN:** Señal de reset del PGA.

Capítulo 6

Software

En el *Qt Creator* queremos conseguir un entorno gráfico que sea intuitivo y fácil de manejar para el usuario final. La aplicación, denominada *SGandO*, siglas de *Signal Generator and Oscilloscope*, será la encargada de enviar los parámetros para la generación de las diferentes señales a través del puerto serie RS–232 del que dispone la *Spartan–3AN Starter Kit* (figura 3.12) cuando el control del mismo lo tenga este programa. Así mismo, se encargará de capturar los datos proporcionados por la FPGA mediante el mismo puerto mencionado anteriormente, que a su vez fueron adquiridos por el ADC, y así poder mostrarlos gráficamente en la pantalla de *SGandO*.

Las clases que forman *SGandO* son:

- *mainwindow*.
- *portwindow*.
- *qcustomplot*.
- *rs232serialport*.

Donde los dos primeros, *mainwindow* y *portwindow*, contienen fichero de formulario. Además de las clases mencionadas en los ítems anteriores, *SGandO* se compone del fichero de cabecera *parametros.h*, el fichero fuente para el arranque de la aplicación *main.cpp* y el fichero de recursos *images.qrc*.

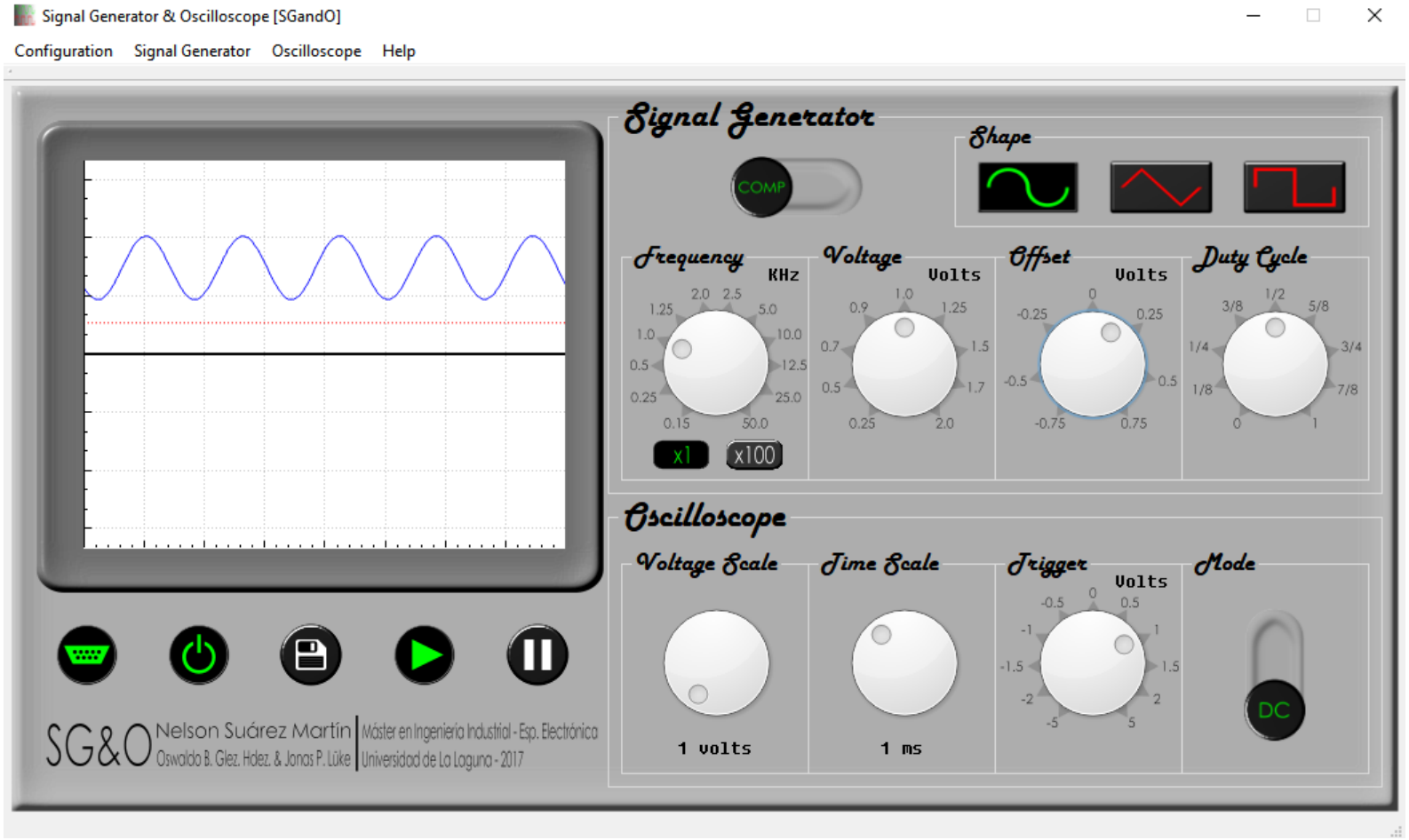


Figura 6.1: Aplicación gráfica para PC *SGandO* (Escala 1:3/4). *Elaboración propia.*

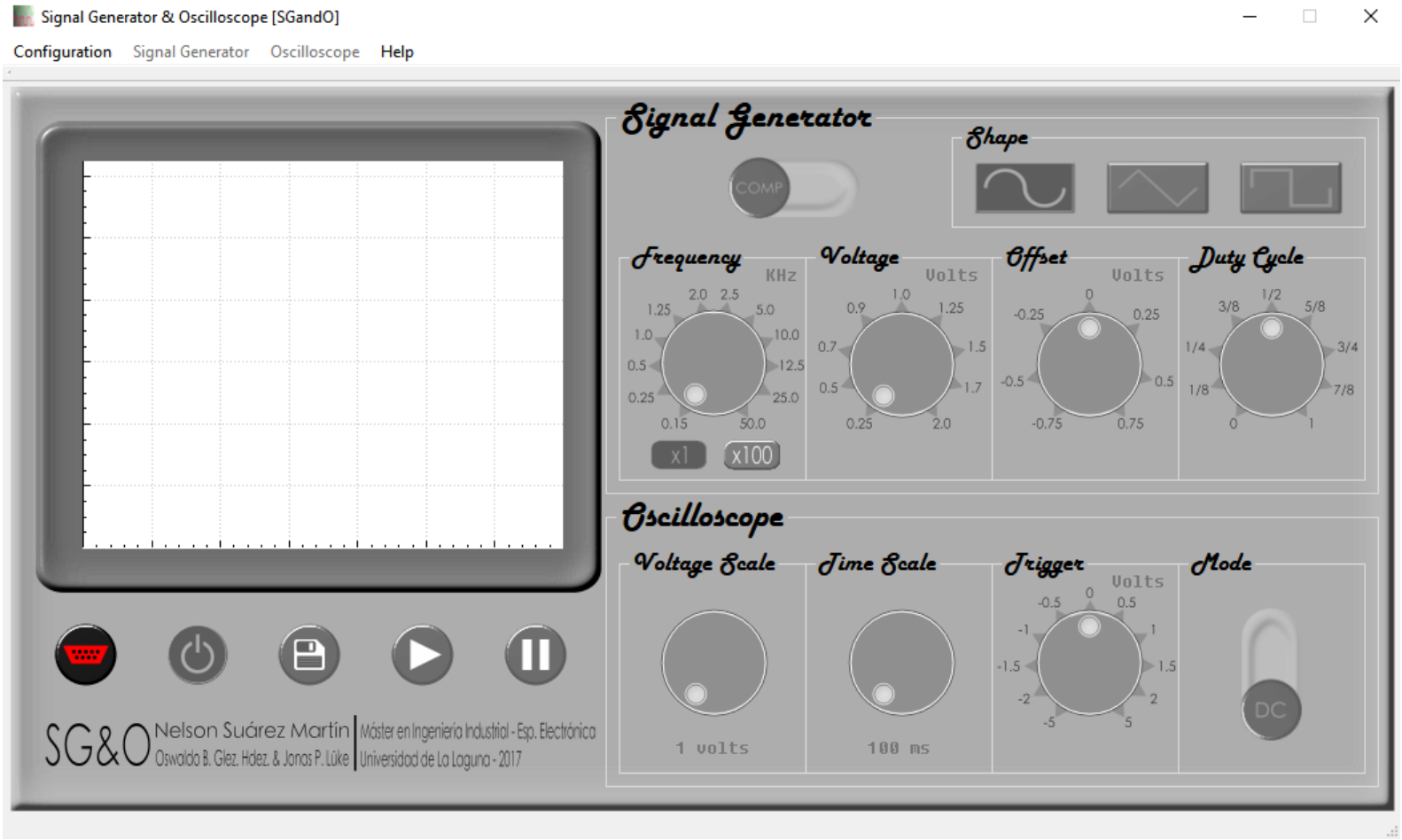


Figura 6.2: Aplicación gráfica para PC *SGandO* no conectado (Escala 1:3/4). *Elaboración propia.*

En la figura 6.1, se muestra la aplicación en funcionamiento y en la figura 6.2 cuando la misma no está conectada a ningún puerto serie.

Exceptuando los diales, el resto de actuadores fueron creados en la aplicación *Inkscape*, vista en apartados anteriores, así como también el fondo de la aplicación y algunos iconos. Todos ellos pueden verse en el anexo a este documento denominado *Objetos en Inkscape*.

6.1. Interfaz Gráfica

La interfaz gráfica creada para la visualización de los datos a modo de osciloscopio y para el manejo de una serie de parámetros para la generación de señales, es la mostrada en la figura 6.1.

Nada más arrancar la aplicación, nos aparecerá un cuadro de diálogo para conectar el puerto serie RS–232 que tengamos disponible en el PC (figura 6.3). Presionando *OK* teniendo un dispositivo de lectura/escritura RS–232 conectado, dará lugar a la imagen de la figura 6.1. En caso contrario, si no hay ningún puerto disponible o haciendo clic en *Cancel*, la aplicación presentará el estado de la figura 6.2.

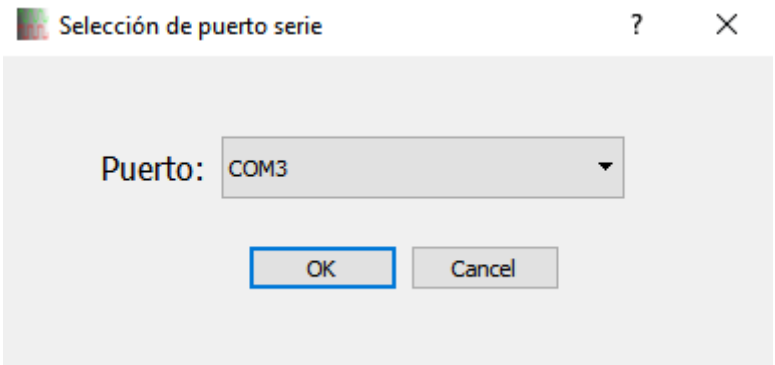


Figura 6.3: Cuadro de diálogo para la selección del puerto serie. *Elaboración propia.*

Situándonos en el segundo de los casos, únicamente tendremos disponible el botón de conectar al puerto serie RS–232, así como a las pestañas de *Configuration* y *Help*, mostradas ambas en la siguiente figura.

Figura 6.4: Pestañas habilitadas cuando *SGandO* está desconectado. *Elaboración propia.*

En la pestaña *Configuration*, si hacemos clic en *Select Port*, se nos abrirá de nuevo el cuadro de diálogo para la selección del puerto serie (figura 6.3). Si hacemos clic en el botón de la aplicación para conectar el puerto serie (figura D.2) o en *Connect RS–232*, la aplicación arrancará como se muestra en la figura 6.1 si antes habíamos seleccionado el puerto o, en caso contrario, aparecerá el mensaje de error de la figura 6.5, lo que dará lugar al presionar *OK* nuevamente la ventana de diálogo para la selección del puerto.

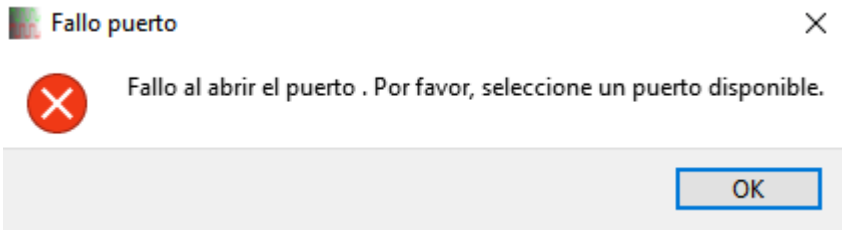


Figura 6.5: Pestañas habilitadas cuando *SGandO* está desconectado. *Elaboración propia.*

La pestaña de *Help* da la opción de realizar dos acciones. En primero lugar, *Qt Webpage* da acceso a la página web principal de *Qt* y, en segunda instancia, *About* muestra el siguiente mensaje sobre la aplicación creada:

Una vez conectada la aplicación al puerto serie, haciendo clic en el botón de encendido y apagado del generador de señales (figura D.3) habilitaremos o deshabilitaremos los actuadores que intervienen en el manejo de este generador. Como resultado, el osciloscopio no mostrará señal alguna. Esta acción también podrá realizarse desde la pestaña de *Signal Generator* en la opción *Disconnect*.

Haciendo clic en el botón de guardado (figura D.4), puede dar lugar a dos opciones. Si el osciloscopio se encuentra actualizando datos, nos aparecerá el mensaje de información de la figura 6.8. En caso de que se encuentre presionado el botón de pause (figura D.6),

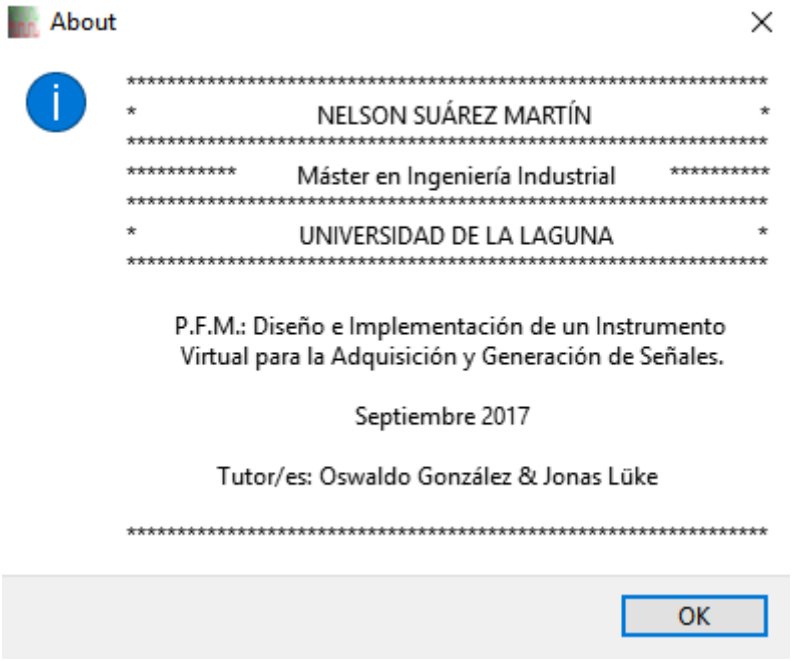


Figura 6.6: Ventana si hacemos clic en la pestaña *Help* → *About*. *Elaboración propia.*

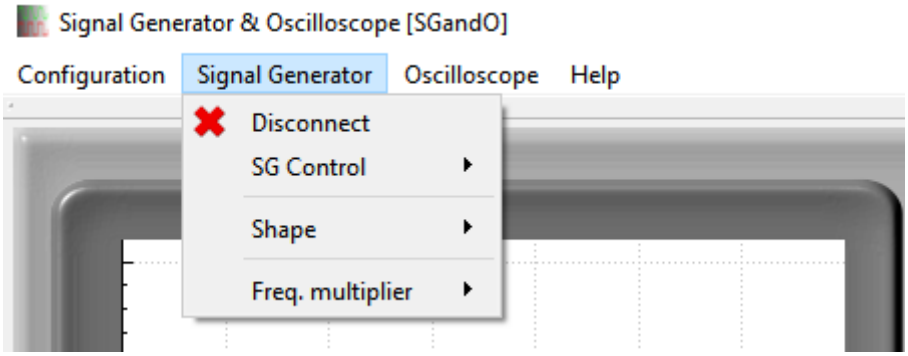


Figura 6.7: Pestañas *Signal Generator*. *Elaboración propia.*

podremos seleccionar la carpeta donde guardar la imagen que se muestra congelada en la pantalla de la aplicación y darle nombre en formato *.PNG* (figura 6.9). Sus dimensiones serán de 800x600 px. Esta acción puede realizar también desde la pestaña *Oscilloscope* → *Save Picture* (6.10).

El siguiente botón siguiendo el orden llevado hasta ahora es el de *Play* (figura D.5), cuya acción es la de tomar datos del puerto serie y representarlos en el display de la aplicación. Esta misma acción puede realizarse desde la pestaña *Oscilloscope* → *Play*

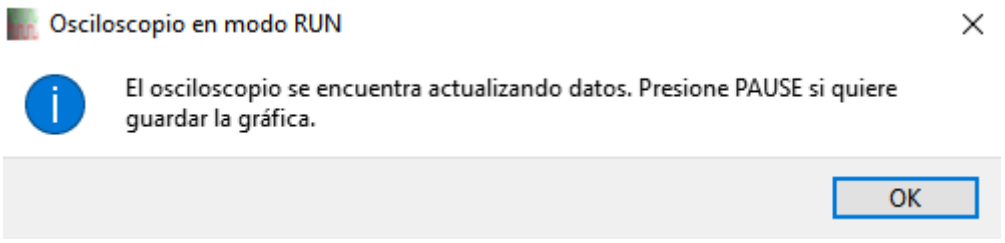


Figura 6.8: Ventana si hacemos clic en *Guardar gráfica como...* si el osciloscopio se encuentra actualizando datos. *Elaboración propia.*

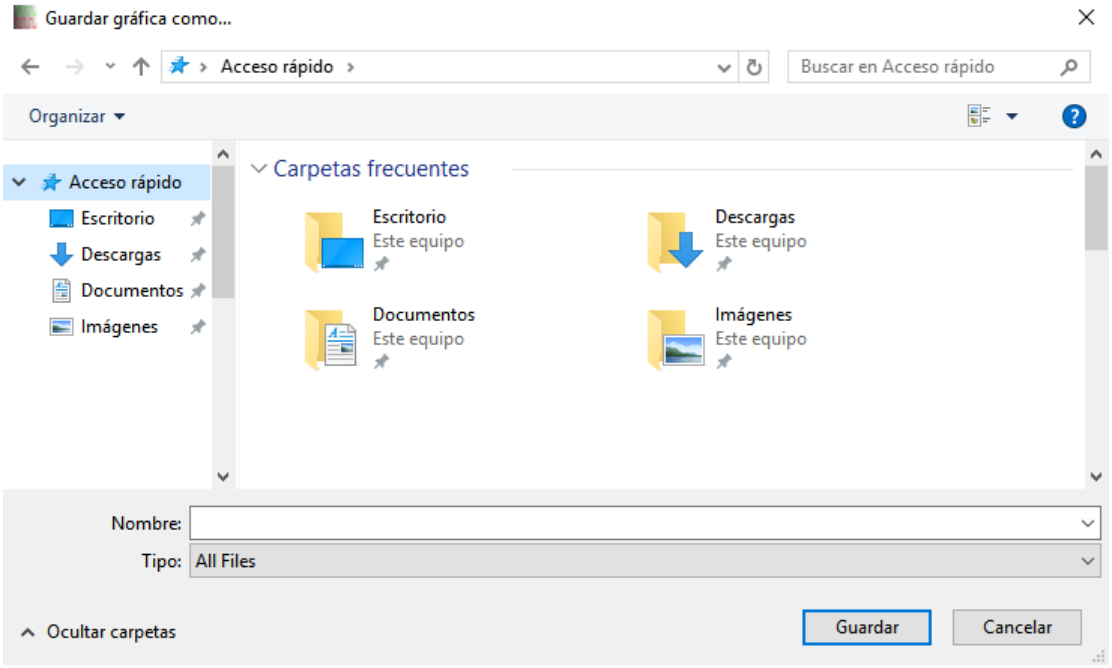


Figura 6.9: Ventana si hacemos clic en *Guardar gráfica como...* si el osciloscopio no se encuentra actualizando datos. *Elaboración propia.*

(figura 6.10). En este último menú, en caso de estar la aplicación actualizando datos, la palabra y el icono de *Pause* sustituirá a la de *Play*.

Haciendo clic en el botón de pause (figura D.6 pararemos la actualización de datos, acción la cual también puede hacerse desde la pestaña *Oscilloscope* → *Pause*. Como ya se comentó anteriormente, en caso de que ya se encuentre presionado el botón de *Pause*, esta opción aparecerá con el icono y el texto de *Play*.

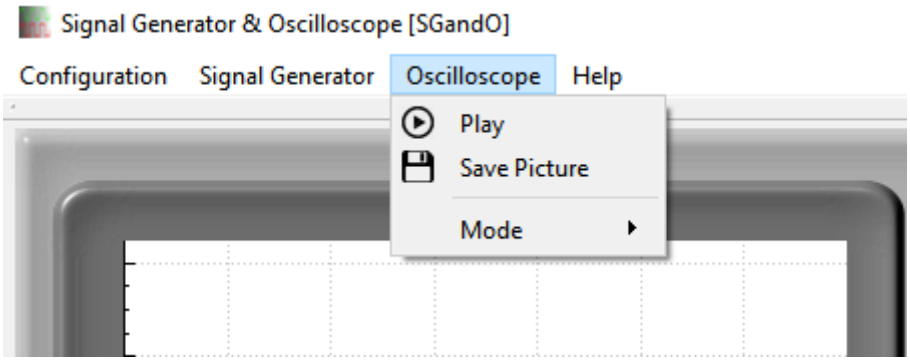


Figura 6.10: Pestaña *Oscilloscope. Elaboración propia.*

6.1.1. Generador de Señales

Centrándonos ahora en el apartado del generador de señales, y como ya se comentó en el apartado del diseño hardware destinado a él, este puede ser controlado desde la aplicación de PC o desde la propia placa que aloja la FPGA. Este control se realiza por medio del selector de control del generador (figura D.7). Cuando el control lo tiene la placa, los mandos de la aplicación que pertenecen al generador de señales se deshabilitan. Como en los casos anteriores, esta acción podrá llevarse a cabo también desde la pestaña *Signal Generator* → *SG Control* → *PC*, en caso de que el control lo esté llevando la placa, o *Board*, en caso de que el control en ese instante lo esté llevando el PC. En este submenú, solo estará una de ellas habilitada.

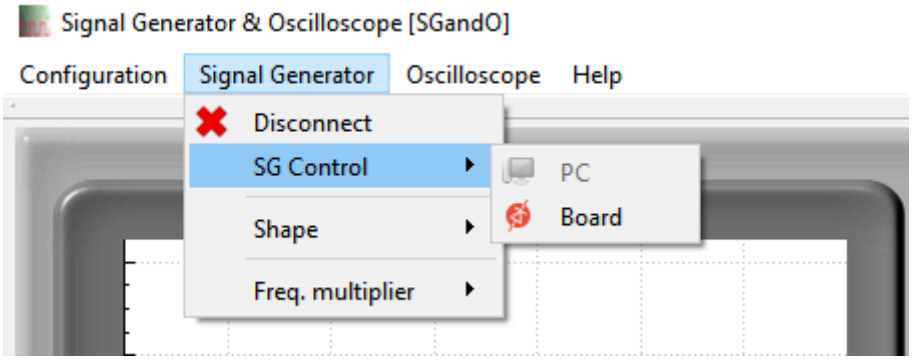


Figura 6.11: Pestaña *Generator Signal* → *SG Control. Elaboración propia.*

En esta aplicación también es posible seleccionar la forma de la señal que queremos

generar, pudiendo seleccionar entre la senoidal, la triangular o la cuadrada (figura D.9). Como ya se comentó en el apartado dedicado al hardware, la opción de señal continua no se implementó en esta aplicación. La forma de la señal también se puede seleccionar desde la pestaña *Signal Generator* → *Shape*, seleccionando una de las opciones que corresponden a cada una de las formas de señal.

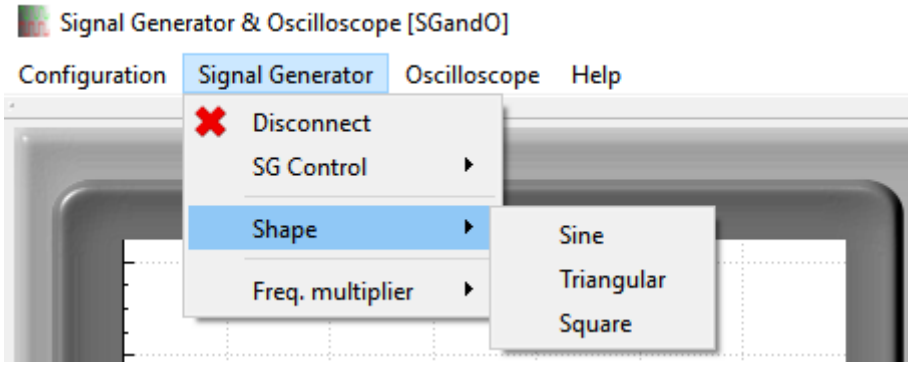


Figura 6.12: Pestaña *Generator Signal* → *Shape. Elaboración propia.*

La parte del generador de señales posee también cuatro diales (figura 6.1) para proporcionar los parámetros que queremos darle a la señal seleccionada. Estos diales son:

- **Frequency:** Para seleccionar la frecuencia en KHz de la señal a generar. Esta frecuencia dependerá de la selección del multiplicador[1]:

  x1: La frecuencia en KHz es la que se muestra en la aplicación alrededor del dial (en esta opción tan solo llega hasta los 10.0 KHz).

  x100: La frecuencia que aparece en la aplicación alrededor del dial es multiplicada por 100 (en esta ocasión están disponibles todas las frecuencias del dial).

- **Voltage:** Posibilidad de seleccionar el voltaje pico a pico de la señal que se desea generar. Las unidades que pueden ser seleccionadas están en voltios, y pueden ser: 0.25, 0.5, 0.7, 0.9, 1.0, 1.25, 1.5, 1.7 y 2.0.

- **Offset:** *Offset* que se le puede añadir a la señal que se quiere generar tomando

[1]Esta acción también podrá llevarse a cabo desde la pestaña *Signal Generator* → *Freq multiplier*

valores que van desde -0.75 voltios hasta los +0.75 voltios en intervalos de 0.05 voltios.

- **Duty Cycle:** Selección del ciclo de trabajo (T_{ON}) de la señal a generar. Las posibilidades son: 0, 1/8, 1/4, 3/8, 1/2, 5/8, 3/4, 7/8 y 1.

6.1.2. Osciloscopio

El apartado del osciloscopio cuenta con tres diales:

- **Voltage Scale:** Voltios x división del display del osciloscopio. La acción de este dial sobre la pantalla es nula. Con su actuación lo que estamos modificando es la ganancia del PGA de la placa realizada para este proyecto, siendo el mínimo de ganancia x1 y el máximo x64.

- **Time Scale:** Unidad de tiempo x división del display del osciloscopio. Actuando sobre este dial, modificamos la longitud de las particiones del eje 'y' del display, además de modificar la señal de reloj de adquisición del ADC instalado en la placa realizada para este proyecto.

- **Trigger:** Nivel del umbral de disparo en voltios del osciloscopio. La actuación de este dial lleva consigo un reseteo de las memorias de adquisición vistas en el módulo en VHDL de *adq_module.vhd*. El *trigger* va desde los -2 voltios hasta los +2 voltios en intervalos de 0.5 voltios, excepto el de mayor y menor valor que es de +5V y -5V. Como se puede apreciar en la figura 6.1, el nivel del *trigger* puede verse en el display del osciloscopio como una línea discontinua en color **rojo**.

- **Mode:** Este selector eliminará el nivel de continua con el que cuenta la señal generada. Su activación también podrá hacerse desde la pestaña *Oscilloscope* → *Mode* (figura 6.10). Esta acción generará un mensaje de información como el mostrado en la figura 6.13 cuando el modo activado sea el AC. En este caso, el dial de *offset* volverá a la posición '0' y no se podrá actuar sobre él siempre y cuando el modo AC se encuentre activo.

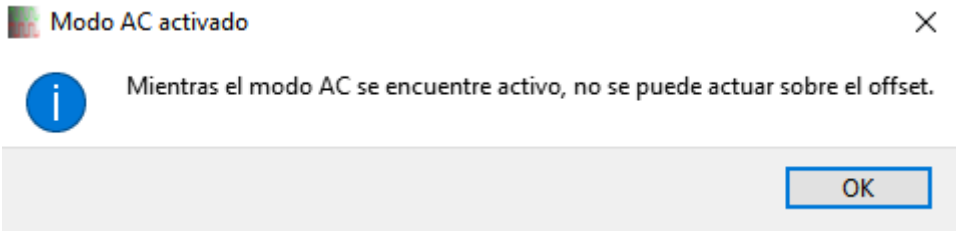


Figura 6.13: Mensaje de información de la activación del modo AC. *Elaboración propia.*

6.2. Comunicación Serie (RS–232)

La clase para hacer posible la comunicación serie fue tomada de la página web de *The Qt Company* [22], a la cual se le cambió el nombre y se le añadieron algunas líneas de código necesarias para la realización del proyecto actual, concretamente las mostradas en el siguiente código pertenecientes al fichero fuente del denominado *rs232serialport.cpp*.

```
1  /* Inicialización y puesta a 0 de la memoria de datos de        *
2   * entrada.                                                    */
3  void rs232serialPort::restart(){
4     k = 0;
5     for(int i = 0; i < memoria.size(); i++){
6        memoria[i] = 0;
7        m_readData[i] = 0;
8     }
9  }
10
11 /* Lectura de los datos del puerto serie.                       */
12 void rs232serialPort::handleReadyRead(){
13
14    m_readData = m_serialPort->readAll();
15
16    for(int i = 0; i < m_readData.size(); i++){
17       if (k < MEMO_DISPLAY){
```

```
18            memoria[k] = m_readData[i];
19            k++;
20            if (k == MEMO_DISPLAY){
21               QByteArray c = memoria;
22               emit dataReady(c);
23            }
24         }
25      }
26   }
```

En el código mostrado anteriormente, la llamada a la función *restart* pondrá a '0' todos los valores contenidos en la memoria de 256 datos. La segunda llamada se realiza para la lectura de los datos por el puerto serie y relleno de la memoria antes mencionada.

El envío de parámetros hacia la FPGA se realiza llamando a la función que se muestra en las siguientes líneas de código:

```
1  /*************************************************************
2   * Envío de parámetros hacia la FPGA.                        *
3   *************************************************************/
4  void MainWindow::send_command(unsigned char b){
5  QByteArray ba;
6  ba[0] = b;
7  rs232.write(ba);
8  rs232.flush();
9  }
```

6.3. Representación Gráfica

La representación gráfica de los datos que entran por el puerto serie se lleva a cabo gracias a la clase creada por *Emanuel Eichhammer*, *qcustomplot*©[23], la cual nos permite una multitud de posibilidades a la hora de llevar a cabo la representación gráfica

de las señales generadas. Entre otros muchísimos aspectos, esta clase nos permite seleccionar el tipo de gráfica, incluir leyendas, personalizar ejes, hacer zoom en la gráfica representada, así como utilizar la opción *drag* para llevar a cabo un arrastre de la misma, etc. En la figura 6.1 se muestra la pantalla de la gráfica utilizada en la aplicación del proyecto actual, *SGandO*.

Capítulo 7

Placa de Circuito Impreso (PCB)

Para el desarrollo de este proyecto se ha llevado a cabo la elaboración de una placa de circuito impreso (PCB) que mejore alguno de los componentes con los que cuenta la placa de *Xilinx, Inc.*, la *Spartan–3AN FPGA Starter Kit* [1]. El diseño de la PCB se realizó con el software libre *KiCad*, aplicación de la que ya se habló en el apartado 4.6 y de la cual se cuenta con una guía práctica para facilitarnos la labor en el diseño e implementación de la placa que queremos realizar [16].

Para elaborar la PCB, lo primero es decidir dónde se conectará la misma en la placa de *Xilinx*. De los puertos disponibles, el único que cumple con un número de entradas y salidas aceptable es el *FX2_Connector* (figura 3.1), cuya disponibilidad de pines es la mostrada en el apartado 3.3 y su disposición puede consultarse en el manual de la placa de *Xilinx* [1]. El conector que se instaló en la placa es el mostrado en la figura 3.10, y cuyo *datasheet* se encuentra en el apartado de los anexos de esta memoria.

A continuación, se decidieron los componentes que iban a ser instalados, separados por dos grupos:

- Generador de señales a partir de los datos proporcionados por la FPGA.

- Adquisición de datos que son enviados y tratados por la FPGA.

7.1. Generador de señales

Hay dos componentes principales en el diseño de esta parte de la PCB: El convertidor digital–analógico o DAC y el amplificador diferencial.

Para el primero de ellos, se decidió instalar el *AD9752* [18] de *Analog Devices, Inc.*, pudiendo generar señales de hasta 5 MHz. Este diseño se llevó a cabo siguiendo el *datasheet* del componente (figura 7.1), el cual está disponible en los anexos del actual documento.

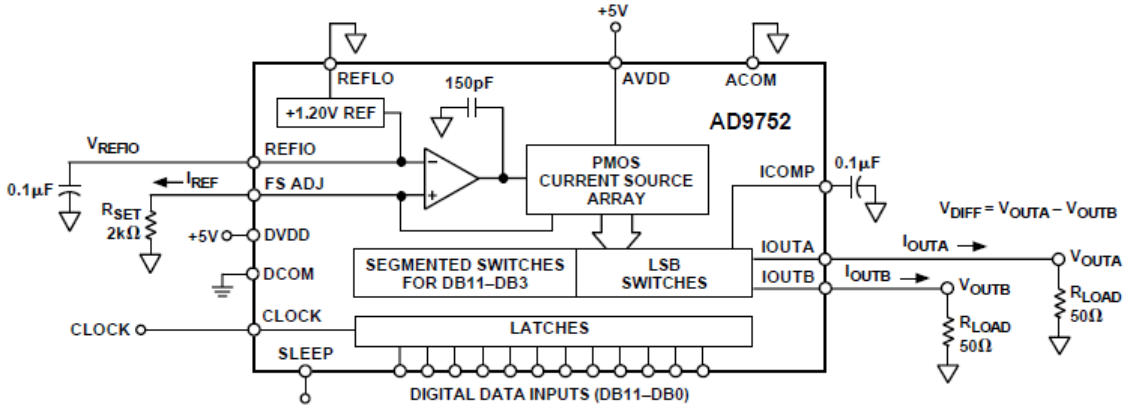


Figura 7.1: Diagrama de bloques del DAC *AD9752* [18].

La salida del DAC se conectó a un amplificador diferencial (figura 7.2), lo que ayuda en la cancelación del error en modo común, así como a el disminuir ruido, distorsiones en la señal y offset.

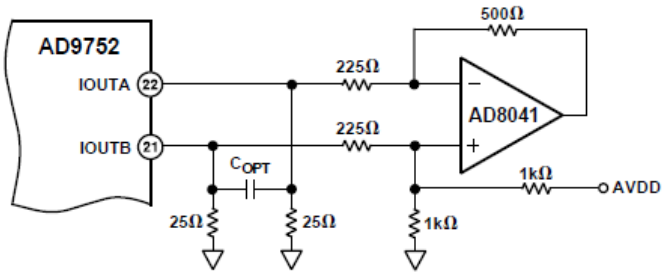


Figura 7.2: Amplificador diferencial a la salida del DAC [18].

En lugar del integrado *AD8041* que aparece en el esquema anterior, por disponibilidad en el laboratorio, se decidió instalar el *OPA354* [25]. El condensador C_{OPT} hace las

funciones de un filtro pasabaja, siendo su valor de $27pF$, dando lugar a una frecuencia de corte de $250MHz$ aproximadamente.

$$C_{OPT} = \frac{1}{2\pi * R * f_c} = \frac{1}{2\pi * 25\Omega * 250MHz} = 25{,}4x10^{-12}F$$

La salida de la señal del amplificador diferencial se lleva a un conector SMA tipo hembra.

7.2. Osciloscopio

En la parte dedicada a la adquisición de datos, ésta, igual que la anterior, se divide en dos componentes principales: el amplificador de ganancia programable o PGA y el convertidor analógico–digital o ADC.

Para el PGA se ha decidido instalar el mismo componente que el instalado en la placa de *Xilinx* pero con una variante. El instalado en la placa *Spartan* es el modelo 6912–1, mientras que el que se puso en el diseño es el 6912–2[21]. La única diferencia radica en que en el primero de ellos las ganancias son: 0, 1, 2, 5, 10, 20, 50 y 100 V/V; mientras que el instalado en el diseño del proyecto actual, estas ganancias son: 0, 1, 2, 4, 8, 16, 32 y 64 V/V.

El esquema que se siguió para su instalación es el mostrado en la siguiente figura, donde la única diferencia respecto al instalado en la PCB del proyecto es que al no usarse el *Canal B*, la entrada del mismo fue conectada a *GND* y la salida al aire. La entrada al PGA se realiza mediante un conector SMA tipo hembra, igual que el montado para la salida del generador de señales.

El ADC utilizado fue el *ADS805* [20], cuya configuración para la señal de entrada es la que se muestra a continuación, siendo la que mayor margen de entrada permite.

Como se puede ver en la figura anterior, la señal de entrada V_{in} debe estar centrada en 2.5 voltios, aspecto a tener en cuenta en caso de introducir señales externas. En el supuesto de que la señal de entrada sea la de salida del generador de señales diseñado en esta misma PCB, no existe ningún problema ya que el amplificador diferencial tiene

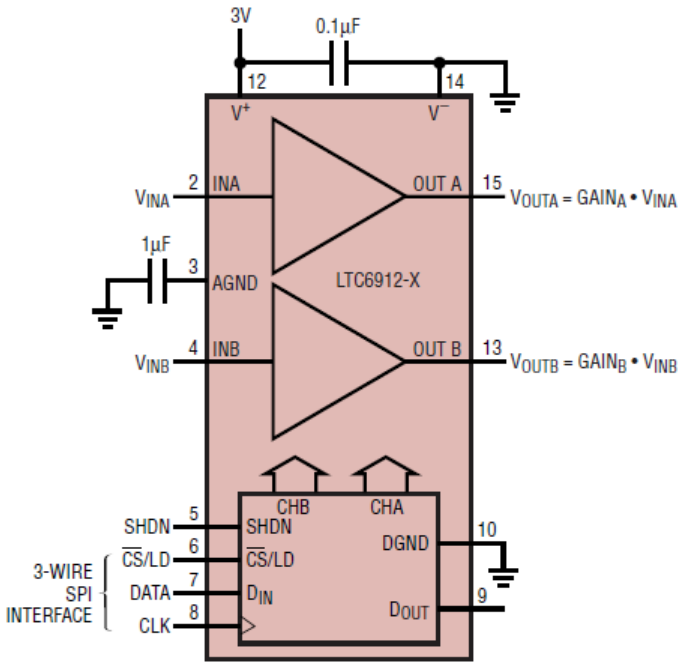


Figura 7.3: Diagrama del PGA instalado en el diseño de la PCB [21].

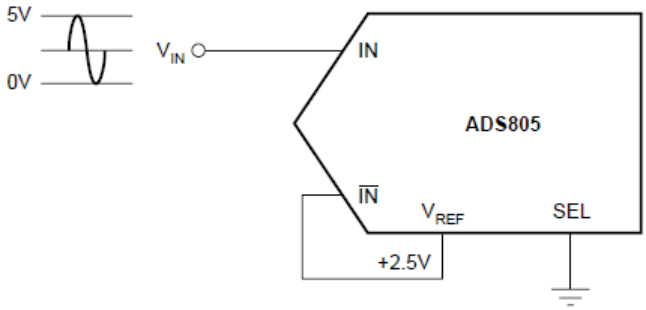


Figura 7.4: Configuración para la señal de entrada al ADC [20].

un *offset* de 2.5 voltios (figura 7.2) . La conexión de ambos puede realizarse gracias a latiguillos donde en los extremos tienen un conector SMA tipo macho (figura 7.5).

7.3. Esquemas PCB

En las siguientes páginas, se encuentran los esquemas electrónicos de la PCB realizada, así como también la PCB propiamente dicha. A continuación, se muestra el modelo tridimensional de la misma.

Figura 7.5: Latiguillo de conexión SMA_macho–SMA_macho. *RS–online.com.*

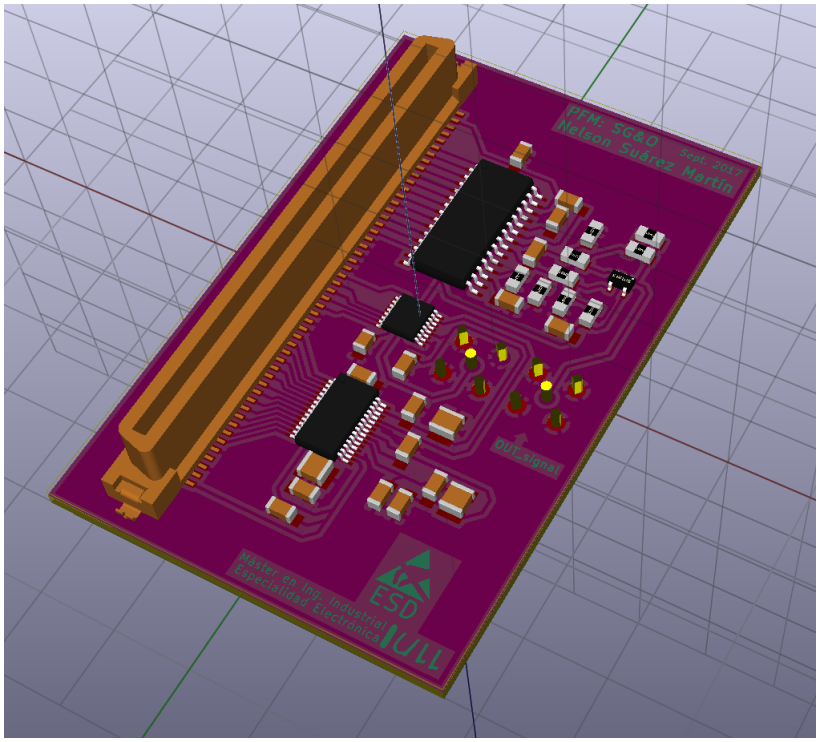


Figura 7.6: Vista frontal de la PCB fabricada. *Modelo 3D extraído de KiCad.*

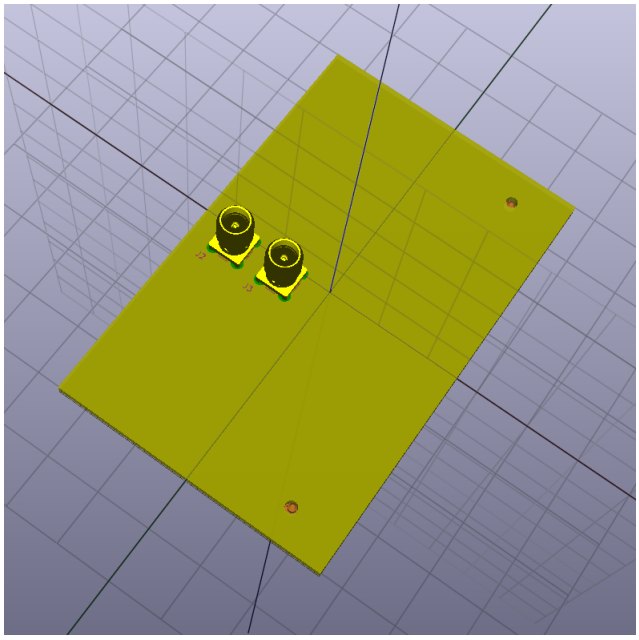


Figura 7.7: Vista frontal de la PCB fabricada. *Modelo 3D extraído de KiCad.*

Parte IV

Resultados y Conclusiones

Capítulo 8

Resultados

A continuación, se exponen los resultados del proyecto que ha sido descrito en esta memoria. Para la comprobación del correcto funcionamiento tanto del generador de señales como del osciloscopio, se ha hecho uso de los siguientes equipos pertenecientes al laboratorio de la Universidad de La Laguna.

- Osciloscopio RIGOL®MSO1104–Z [26].

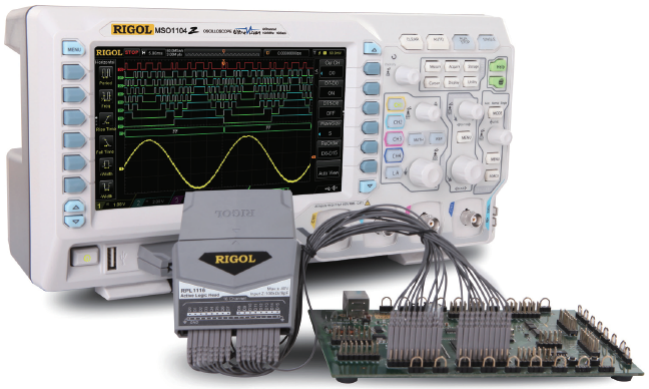


Figura 8.1: Osciloscopio RIGOL MSO1104–Z [26].

- Generador de funciones PROMAX GF–232 [27].

La placa final que se realizó con productos y equipos del laboratorio de la Universidad de La Laguna es la mostrada en la imagen de la figura 8.3. La poca disponibilidad de

Figura 8.2: Generador de Funciones PROMAX GF–232 [27].

conectores *FX–2* de *Hirose* no ha posibilitado la realización de una placa definitiva sin los fallos estéticos de la actual. Los problemas surgidos durante la realización de esta placa y otros que han ido aconteciendo se expondrán en el apartado de conclusiones (apartado 9) de esta memoria.

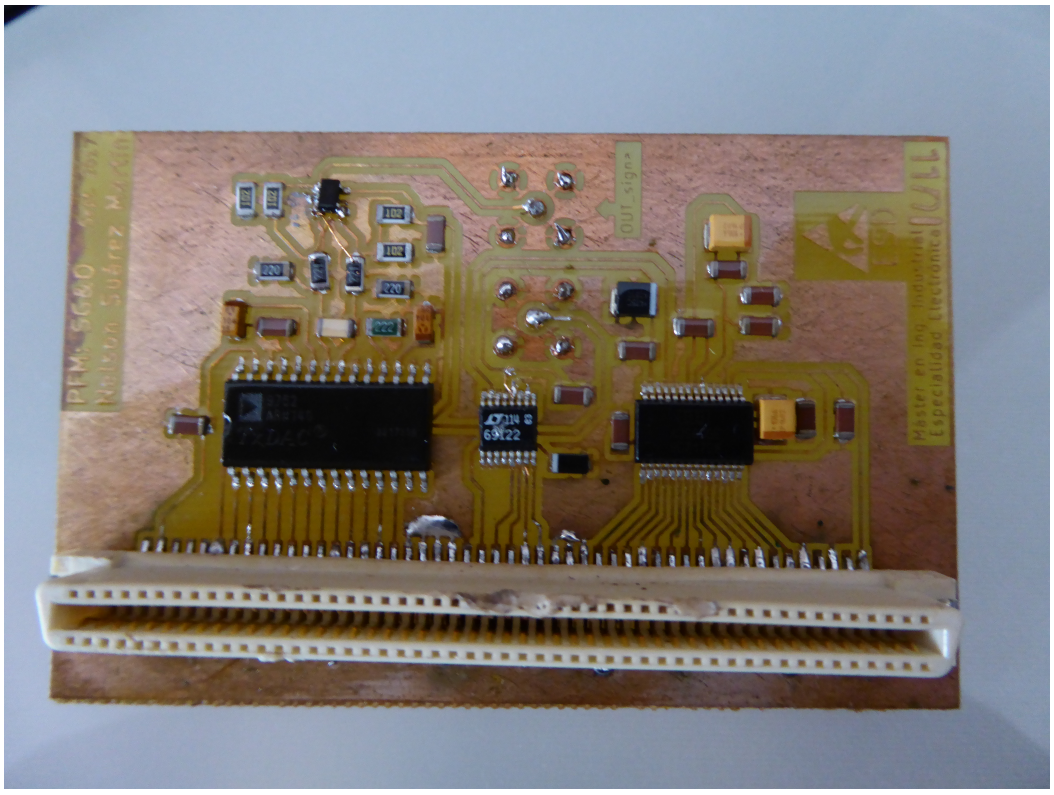


Figura 8.3: Placa realizada para el proyecto actual. *Elaboración propia.*

8.1. Generador de señales

El montaje es el que se muestra en la siguiente imagen tomada cuando se realizaban las pruebas.

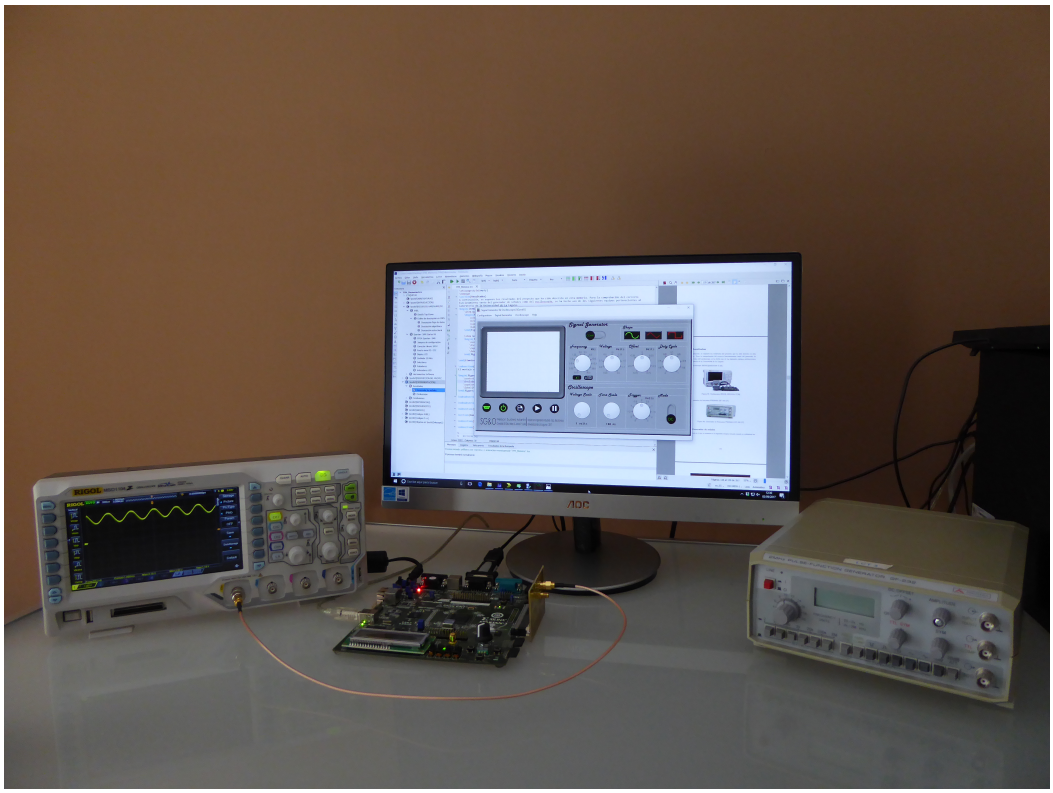


Figura 8.4: Prueba del generador de señales. Vista general. *Elaboración propia.*

8.1.1. Control desde el PC

La imagen anterior fue tomada cuando el generador de señales era controlado desde la aplicación para PC *SGandO*. Como ya se comentó en el apartado 5.7.3, la *Spartan–3AN Starter Kit* presenta la imagen de la figura 8.5 cuando el control del generador de señales lo tiene el software.

La aplicación *SGandO* está configurada como se muestra en la figura 8.6, donde los parámetros enviados a la FPGA son:

- Tipo: Señal senoidal.

- Frecuencia: 1 KHz.
- Voltaje: 1 V.
- Offset: 0 V.
- Ciclo de trabajo: 50 %.

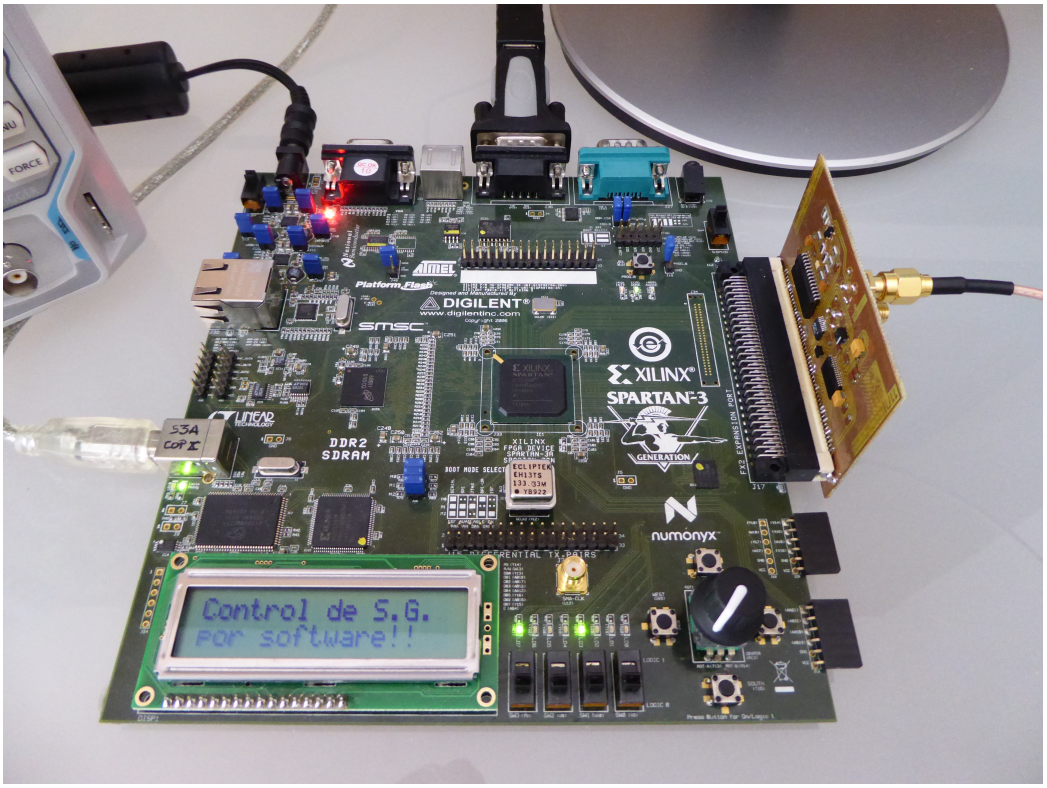


Figura 8.5: Placa *Xilinx* controlada desde el PC. *Elaboración propia.*

Con la imagen capturada del osciloscopio de *RIGOL* 8.7 podemos comprobar que la señal es, efectivamente, una señal senoidal, con una frecuencia aproximada de 1 KHz y 1 V_{pp} y un ciclo de trabajo del 50 %. El offset que presenta se debe al ya comentado en el apartado 7.2 de 2.5 voltios; el añadido por *SGandO* es de 0 voltios.

En las siguientes imágenes podemos ver la generación de las otras dos señales. La primera de ellas, la señal triangular con los mismos parámetros que la señal senoidal

Figura 8.6: Aplicación *SGandO* para la prueba del Generador de Señales. *Elaboración propia.*

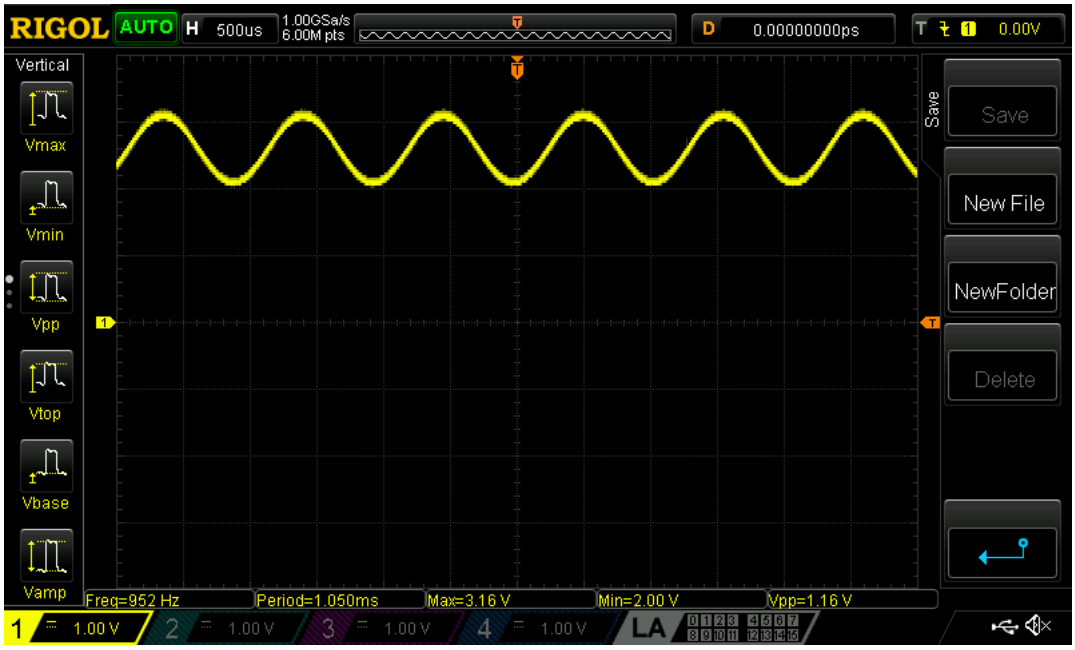


Figura 8.7: Señal de prueba en el osciloscopio de *Rigol*. *Imagen extraída de Rigol Oscilloscope.*

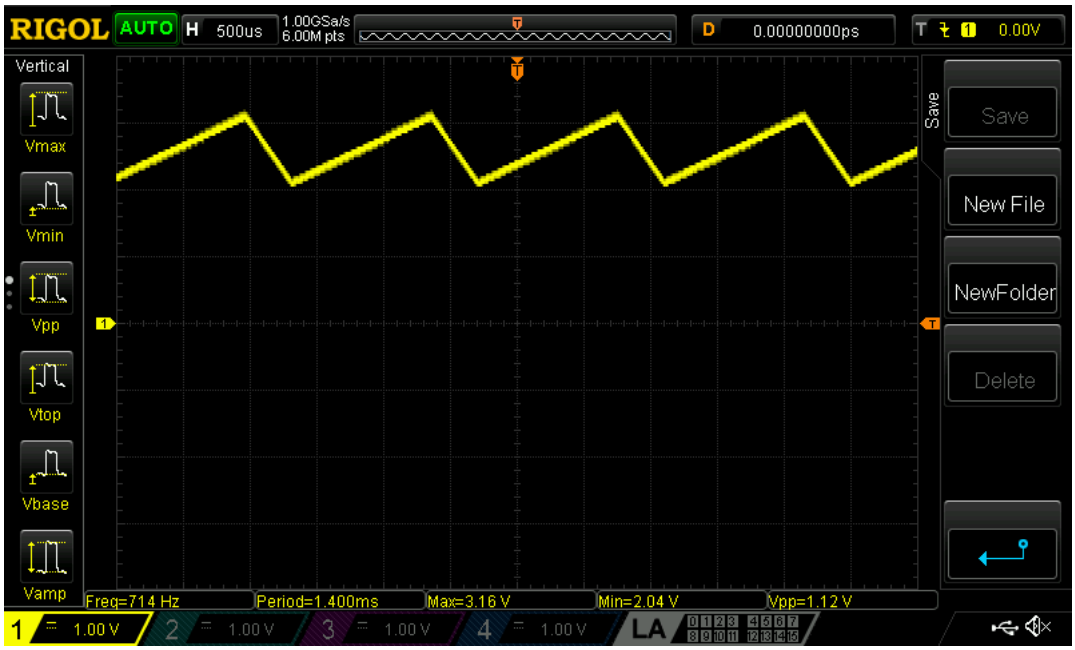


Figura 8.8: Señal triangular con $D = 75\,\%$. *Imagen extraída de Rigol Oscilloscope.*

anterior pero con un ciclo de trabajo el 75 % (figura 8.8). Mientras que en la segunda (figura 8.9), se genera una señal cuadrada con los mismos parámetros que la señal senoidal con el único cambio en el offset, siendo éste de +0.5 voltios.

8.1.2. Control desde la placa

Si el control del generador de señales se lo pasamos a la placa, la imagen de la misma pasa a ser la siguiente:

Donde ahora *SGandO* presentará la forma de la figura 8.11. Desde el primer selector, de izquierda a derecha, se cambia la escala del generador de señales. El cambio de tipo de señal (senoidal, triangular, cuadrada o señal continua) se realiza con los dos último selectores en la configuración ya vista en el apartado 5.1. El pulsador situado más al sur, es el reset de todos los módulos de la FPGA y, el interruptor rotativo, permite la modificación de cada uno de los parámetros. La selección de cada parámetro se realiza presionando el interruptor rotativo. Esta acción se puede ver en la figura 8.10, donde el parámetro a configurar con el interruptor rotativo es el offset de la señal, marcado con

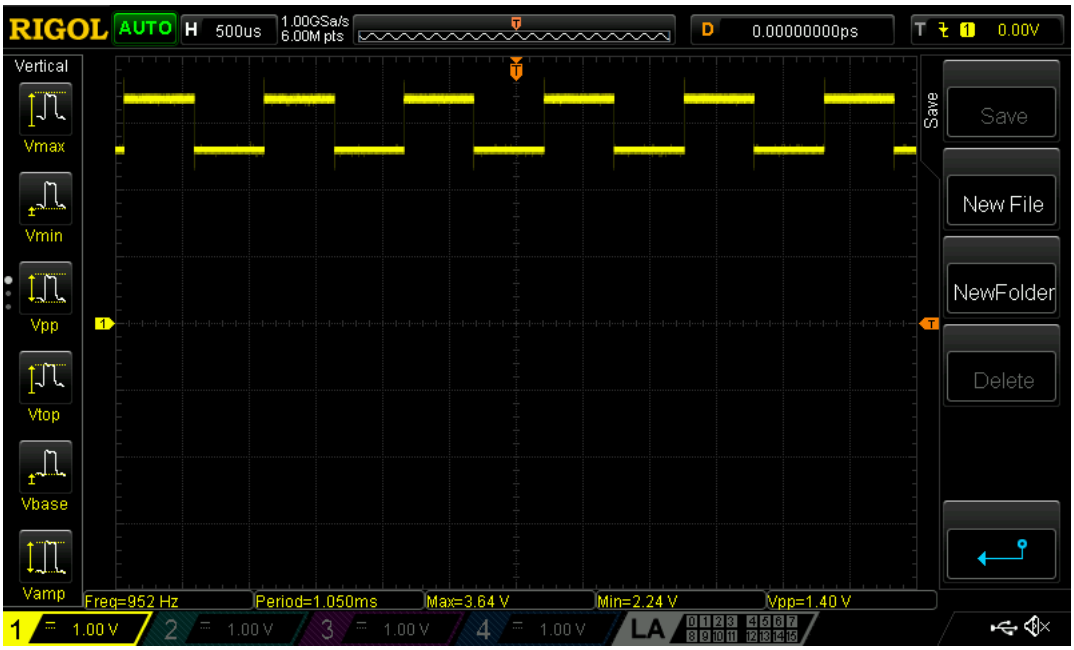


Figura 8.9: Señal cuadrada con $offset = 0{,}5V$. *Imagen extraída de Rigol Oscilloscope.*

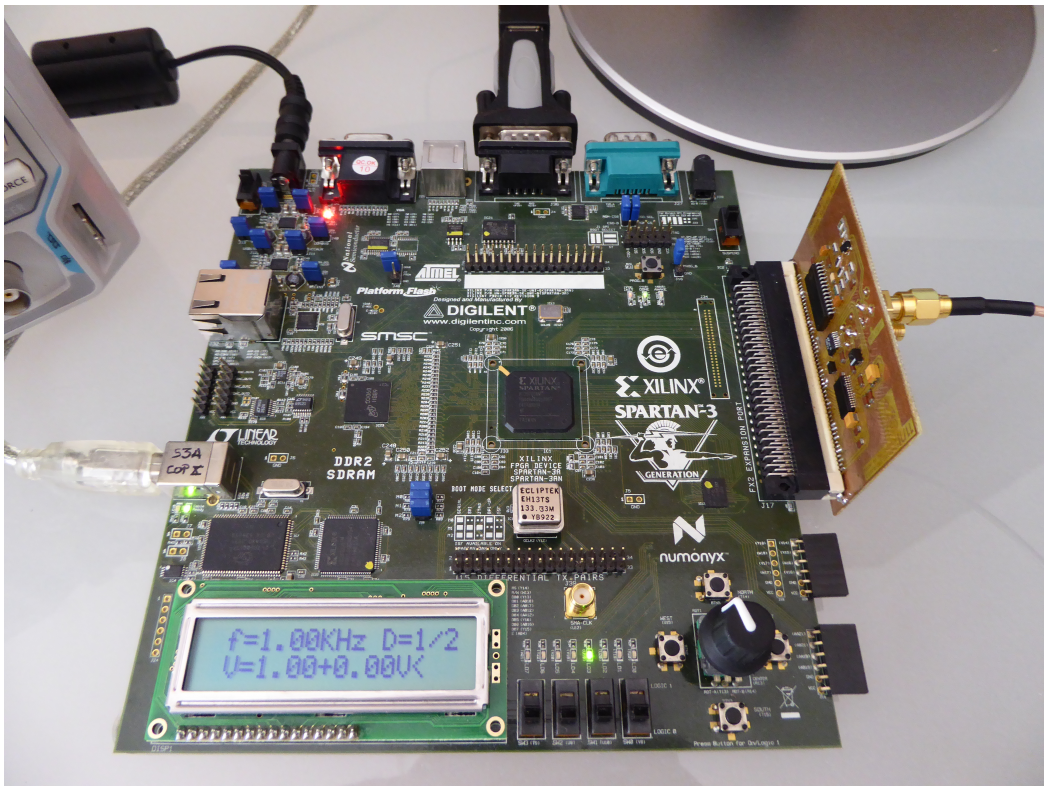


Figura 8.10: Placa *Xilinx* controlada desde la propia placa. *Elaboración propia.*

el símbolo '<'.

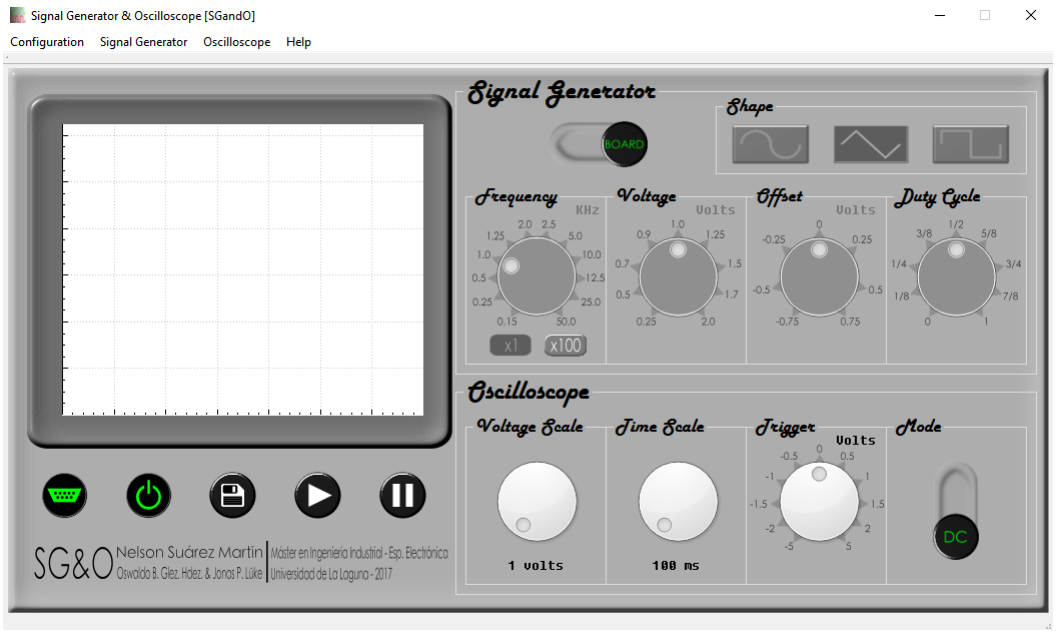


Figura 8.11: Generador de señales controlado desde la placa. *Elaboración propia.*

8.2. Osciloscopio

Para comprobar el correcto funcionamiento del osciloscopio presente en *SGandO*, se ha hecho uso del generador de funciones *PROMAX GF–232* (figura 8.2) antes de probarlo con el propio generador de señales de la FPGA (figura 8.12).

La señal de salida del generador de funciones es una señal del tipo senoidal, de 1 KHz de frecuencia, con una amplitud de 1 V_{pp} y un offset de 2.5 voltios, que está centrado en el margen de entrada del ADC instala en la PCB realizada, que recordemos es de 0–5 voltios.

Esta señal que introduciremos en la FPGA siendo muestreada antes por el ADC de la PCB, es la vista en la captura de pantalla del osciloscopio de *RIGOL* de la figura 8.13.

Utilizando la opción de guardado del display de *SGandO* (apartado 6.1), comprobamos la adquisición de datos del ADC instalado en la PCB fabricada. En esta imagen y en las siguientes capturadas desde el software se ven algunos *glitches* provocados por

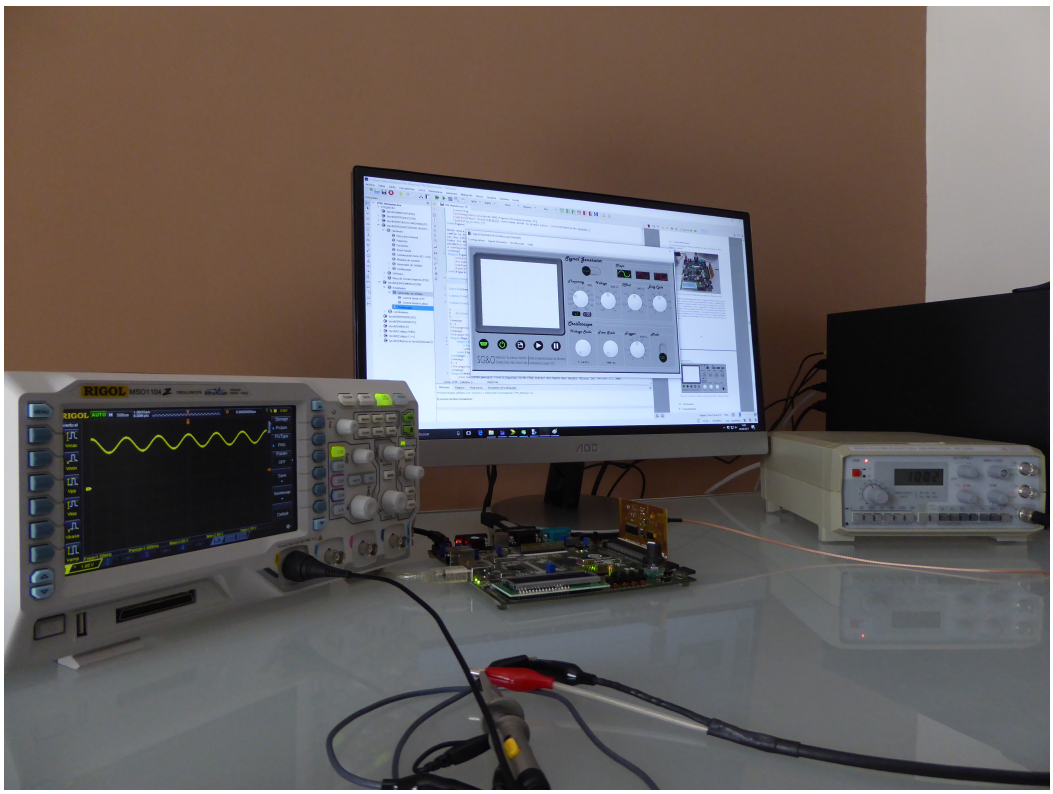


Figura 8.12: Comprobación del funcionamiento del osciloscopio. *Elaboración propia.*

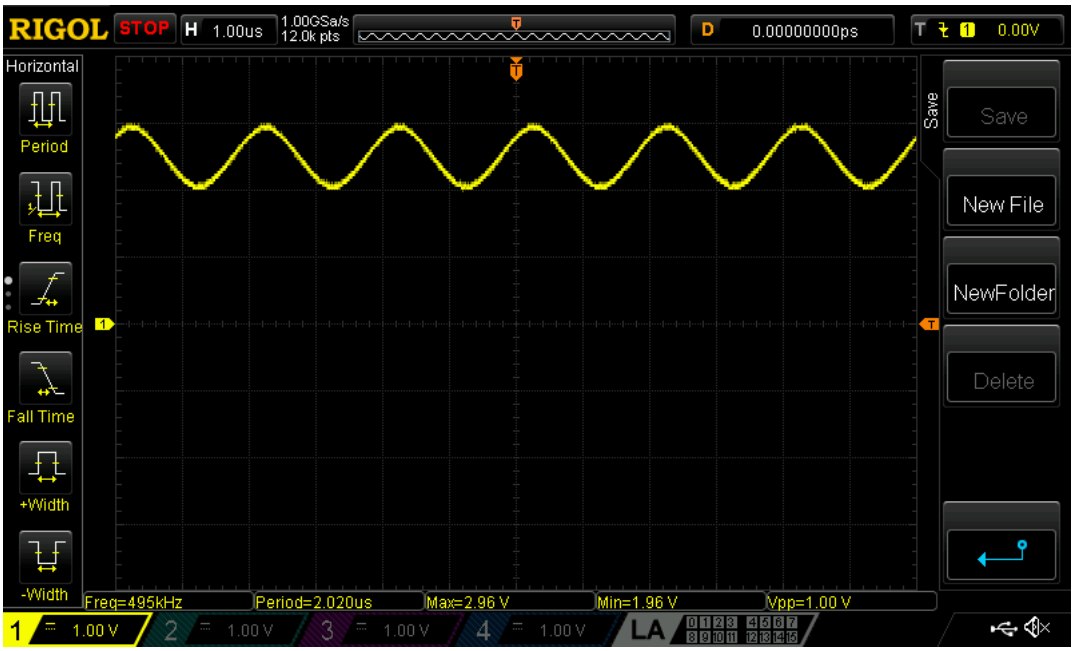


Figura 8.13: Señal de salida del generador de funciones *PROMAX GF–232. Imagen extraída de Rigol Oscilloscope.*

la variación de estados a la salida del vector de datos del ADC. Las imágenes de las diferentes señales son a la misma frecuencia, amplitud y offset.

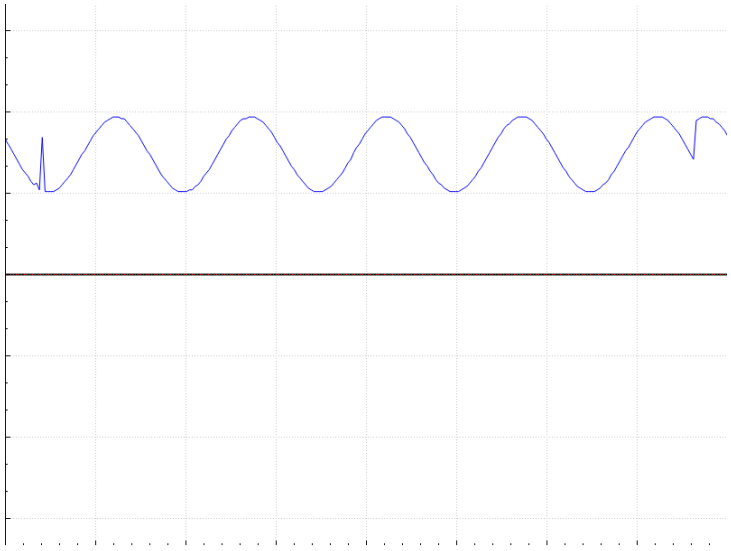


Figura 8.14: Adquisición de datos de una señal senoidal. *Elaboración propia.*

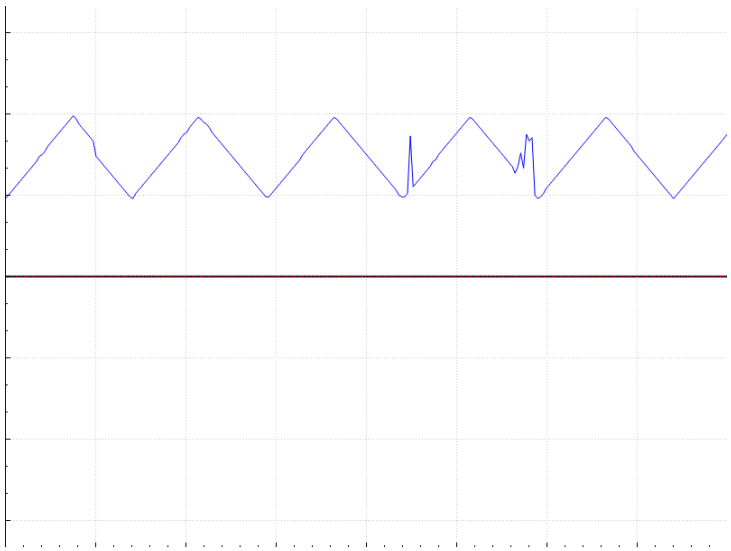


Figura 8.15: Adquisición de datos de una señal triangular. *Elaboración propia.*

Si aplicamos el modo AC del software (figura D.8), podemos ver como se elimina el offset de 2.5 voltios que hemos establecido en el generador de funciones desde un principio (figura 8.17).

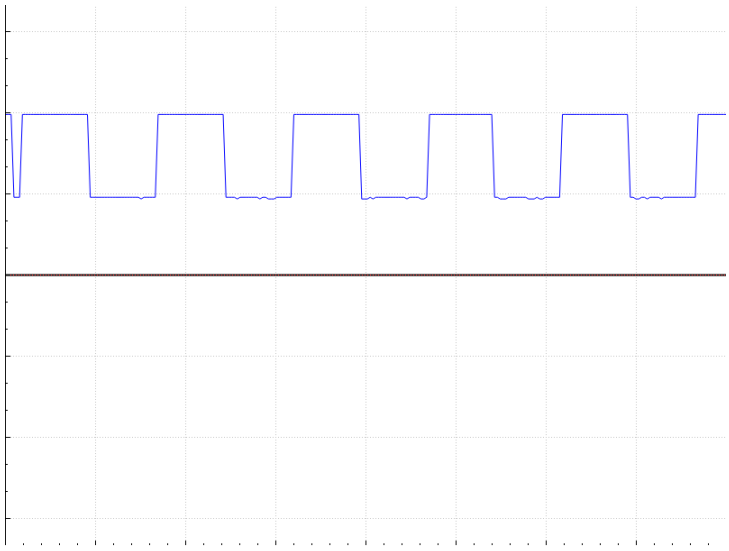


Figura 8.16: Adquisición de datos de una señal cuadrada. *Elaboración propia.*

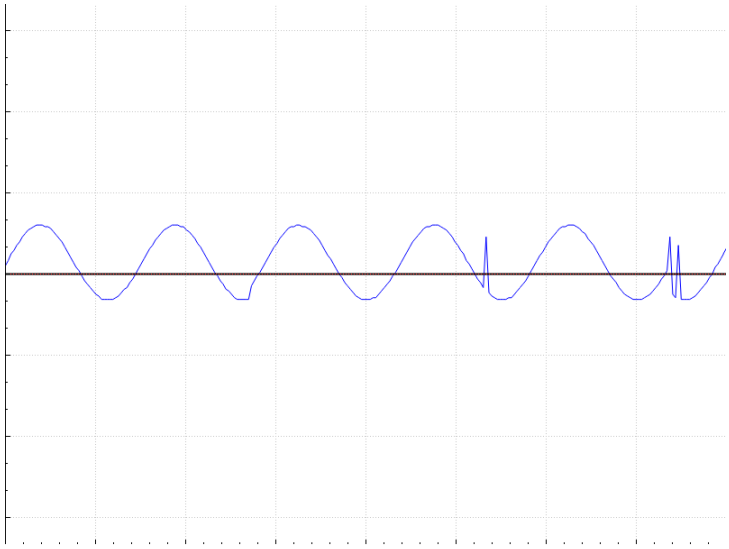


Figura 8.17: Señal de la figura 8.14 en modo AC. *Elaboración propia.*

Las siguientes imágenes corresponden a la conexión de la salida del generador de señales de *SGandO* con el osciloscopio también de *SGandO*.

En las siguientes imágenes capturadas de *SGandO* se comprueba el funcionamiento del *trigger*. Esto se ha hecho modificando la escala de voltajes, es decir, actuando sobre la ganancia del PGA. Esta escala es independiente a la del *trigger*, la cual no se ve modificada ante estos cambios, sino que mantiene siempre los mismos valores, en este

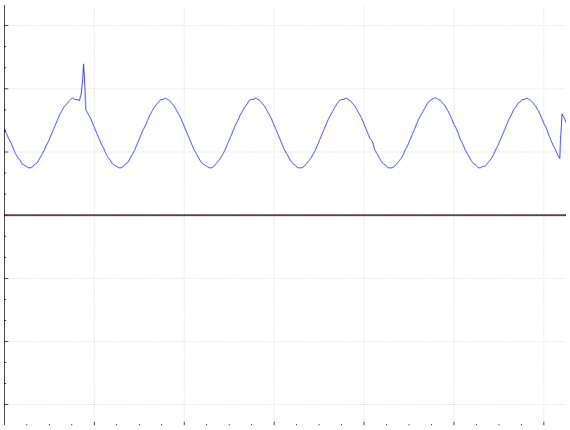


Figura 8.18: Señal senoidal en *SGandO*. *Elaboración propia*.

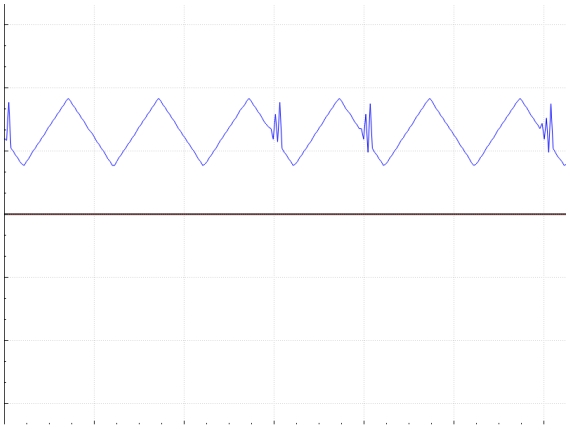


Figura 8.19: Señal triangular en *SGandO*. *Elaboración propia*.

Figura 8.20: Señal cuadrada en *SGandO*. *Elaboración propia*.

caso, +2V según indica el dial del umbral de disparo.

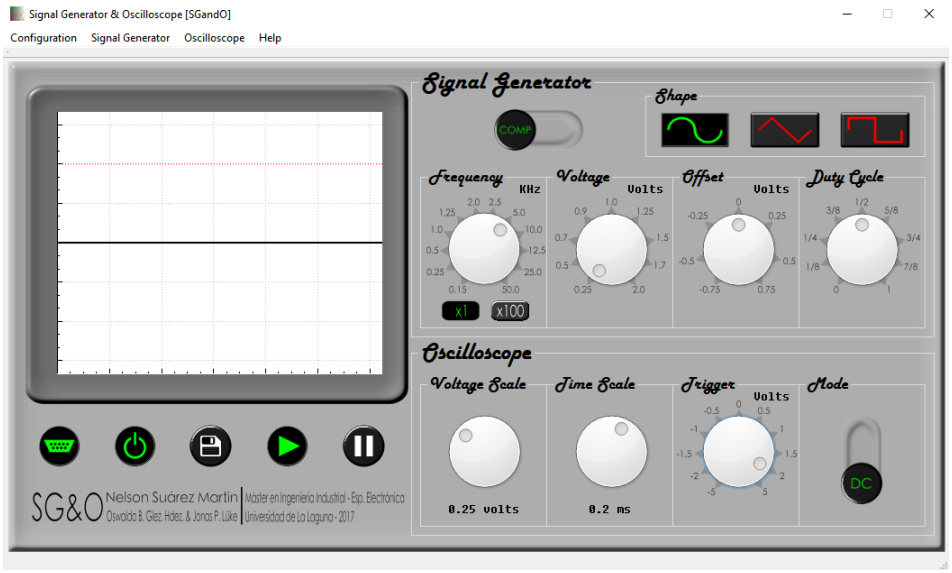


Figura 8.21: A. Prueba de *trigger* con *SGandO*. *Elaboración propia.*

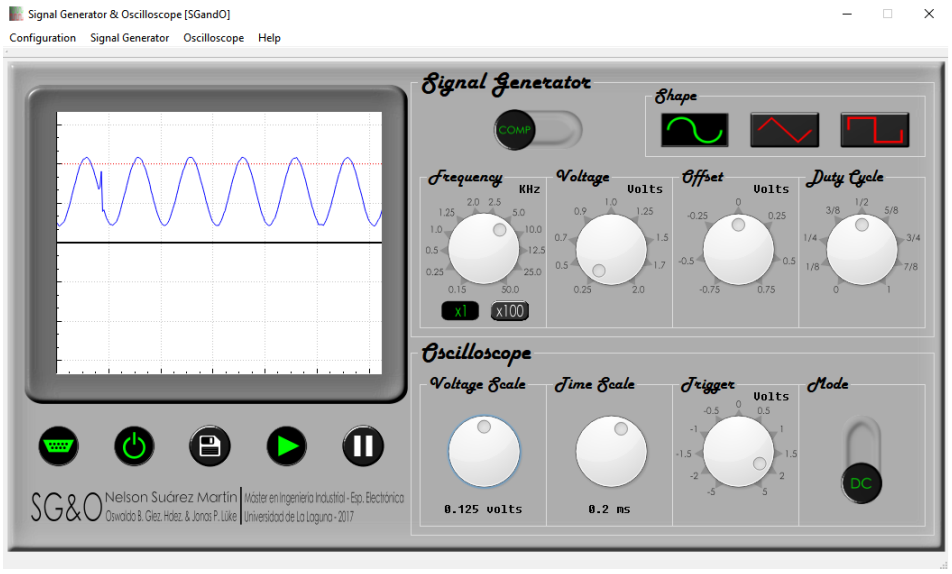


Figura 8.22: B. Prueba de *trigger* con *SGandO*. *Elaboración propia.*

Capítulo 9

Conclusiones

9.1. Conclusiones/*Conclusions*

La realización de este proyecto ha seguido las pautas marcadas en la planificación del mismo (apartado 1.3), con problemas que han surgido en el trayecto pero que finalmente se han solventado, cumpliendo con los objetivos marcados desde el principio.

Teniendo en cuenta la complejidad de los lenguajes de descripción hardware y de programación software, la parte que resultó ser la más compleja fue la realización de la PCB, considerando la poca experiencia en esta materia. En este punto es donde se tiene que destacar la figura de *D. José Carlos Sanluis Leal*, Investigador del Departamento de Ingeniería Industrial de la Universidad de La Laguna, y que sin sus conocimientos no habría sido posible realizar la PCB para este proyecto.

Teniendo experiencia previa en el lenguaje de descripción hardware VHDL, la ayuda de un programa de simulación como es el *ISim* (apartado 4.2), permite que su depuración sea rápida, como ocurre también en la programación orientada a objetos de *C++* que se ha utilizado en la parte software; algo que no tiene cabida dentro de la realización de la PCB. Aún existiendo programas para la simulación de circuitos electrónicos, cuando se llevan a la realidad estos diseños entran en juego muchos factores que hacen que la placa no funcione como se esperaba o simplemente no funcione. Esto fue lo que retrasó en buena medida la finalización esperada del proyecto en la planificación del mismo.

Una vez realizado el primer prototipo de la PCB, un fallo en uno de los bits más significativos del DAC daban como resultado una señal defectuosa. Esto unido a que el PGA no respondía a los cambios de ganancia que se le enviaban desde la FPGA, hicieron que repitiéramos la fabricación de la placa.

El segundo prototipo fue el definitivo (figura 8.3), logrando que ni el DAC ni el ADC tuvieran problemas en sus bits más significativos. Algunos errores en la generación de las señales, llevaron a tener que sustituir el amplificador diferencial, el cual tenía un encapsulado diferente. Así mismo, se tuvieron que re-conectar algunas líneas del PGA, debido a que en la elaboración de la placa el ácido no actuó como debía.

El último obstáculo fue el deterioro del segundo conector *FX–2*, debido a que la altura de la placa dentro del horno hizo que se sobrepasaran las temperaturas máximas marcadas en el perfil del conector. Este hecho dio lugar a que se tuviera que desoldar el conector del primer prototipo para su instalación en este segundo, todo ello realizado a mano.

Una vez resueltos todos los inconvenientes, la placa funciona correctamente según lo desarrollado en el apartado 8, dentro del capítulo de *Experimentación*.

El generador de señales nos permite crear formas tales como senoidal, triangular y cuadrada, a unas frecuencias que van de los 150 Hz a los 5 MHz, con amplitudes de voltaje pico a pico de 0.25 a 2 voltios. Así mismo, se le puede añadir un offset que oscila entre los -0.75 y los +0.75 voltios, y un determinado ciclo de trabajo. Una de las características principales del generador de señales creado es que su funcionamiento puede ser independiente a la utilización del software, ya que desde la placa de *Xilinx* podemos manipular los mismos parámetros que desde el PC.

El osciloscopio permite visualizar señales con parámetros como los del generador comentado en el párrafo anterior, teniendo en cuenta que el margen de entrada se sitúa entre 0 y 5 voltios. La amplificación de la señal no se realiza por medio del *zoom* del software, sino que se hace a través del cambio en la ganancia del PGA, eliminando las distorsiones que generan estos *zoom* digitales. La escala de tiempos se realiza cambiando la frecuencia de muestreo del ADC. Las funciones *trigger* y el modo AC/DC comple-

mentan al osciloscopio de la aplicación denominada *SGandO. The implementation of this project has followed the guidelines set in the planning of the project (section 1.3). Several problems have arisen during the project development but they have finally been solved, fulfilling the objectives aimed at the beginning.*

Considering the complexity of the hardware description languages and software programming, the part that has been proved to be the most complex was the fabrication of the PCB, which was due to my little experience in this matter. In this point it is necessary to emphasize the person Mr. José Carlos Sanluis Leal, Researcher of the Department of Industrial Engineering of the University of La Laguna. Without his knowledge, it would not have been possible to make the PCB for this project.

Having previous experience in the VHDL hardware description language, the help of a simulation program such as ISim (section 4.2) allows for its rapid debugging. The same can be said for the C++ object-oriented programming that has been used in the software, which has no place within PCB development. Although there are programs for the simulation of electronic circuits, when these designs are brought to reality, many factors come into play that prevent the board from working as expected or just not working. This was what delayed the expected project end date.

Once the first prototype of the PCB was made, a failure in one of the most significant bits of the DAC resulted in a faulty signal. This, together with the fact that the PGA did not respond to the gain configurations, caused us to repeat the fabrication of the board.

The second prototype was the definitive one (figure 8.3), making that neither the DAC nor the ADC had problems in their most significant bits. Some errors in the generation of the signals led to have to replace the differential amplifier, which had a different encapsulation. Likewise, some lines of the PGA had to be reconnected, because in the elaboration of the board the acid did not act as it should.

The last obstacle was the deterioration of the second FX–2 connector, because the height of the board inside the oven caused that the marked maximum temperatures in the profile of the connector were exceeded. This fact resulted in the need to desolder the connector of the first prototype for installation in this second, all done by hand.

Once all the problems have been solved, the board works correctly as developed in chapter 8.

The signal generator allows us to create waveforms such as sinusoidal, triangular and square, at frequencies ranging from 150 Hz to 5 MHz, with peak-to-peak voltage amplitudes from 0.25 to 2 volts. Also, you can add an offset that oscillates between -0.75 and +0.75 volts, and a certain duty cycle. One of the main characteristics of the created signal generator is that its operation can be controlled from the board of Xilinx or by the software installed at the PC.

The oscilloscope allows us to display signals with parameters such as the generator mentioned in the previous paragraph, taking into account that the input range is between 0 and 5 volts. The amplification of the signal is not done by means of the zoom of the software, but it is made through the change in the gain of the PGA, eliminating the distortions that generate these digital zooms. The time scale is performed by changing the sample rate of the ADC. The trigger functions and the AC/DC mode complement the oscilloscope developed in the application called SGandO.

9.2. Propuestas de mejora/*Future work*

Como mejoras al proyecto actual, se podrían añadir algunas funciones a la aplicación *SGandO* como son el cálculo de la *FFT* de la señal mostrada en el display del mismo, o mejorar el apartado de la actualización de datos, ya que actualmente puede sufrir saturación ralentizando la aplicación.

Otro de los puntos de mejora se encuentra en el sistema de adquisición de datos en la FPGA, pudiendo ser mejorado para obtener una señal más limpia en la pantalla de nuestro PC. Esto implicaría realizar el proyecto con una FPGA con más recursos que permita memorias de adquisición de datos de mayor longitud. Así mismo, para realizar esto último y lo comentado en el párrafo anterior, sería conveniente el uso de otros protocolos de comunicación más rápidos, como puede ser el USB.

Para terminar, una mejora más avanzada es que este proyecto sea totalmente in-

dependiente de un PC. La PCB podría utilizar un MPSoC como las *Zynq* de *Xilinx, Inc.* que incluyan toda la parte hardware y software, dispositivos que incluso contienen internamente una GPU para el manejo de gráficos.

As improvements to the current project, some functions could be added to the SGandO application such as the calculation of the FFT of the signal shown in the display, or improve the section of the data update, as it can currently suffer certain processing saturation leading to a slowing of the application operation.

Another of the improvement points is in the data acquisition system in the FPGA, and can be improved to obtain a cleaner signal on the screen of our PC. This would imply to implement the project with an FPGA with more resources which provides more memory size for the acquisition of data of greater length. Also, in order to perform this last and the mentioned in the previous paragraph, it would be convenient to use other faster communication protocols, such as USB.

To conclude, a more advanced improvement is that this project is totally independent of a PC. The PCB could use an MPSoC such as Xilinx's Zynq, which includes all hardware and software, devices that even internally contain a GPU for handling graphics.

Capítulo 10

Presupuesto

REFERENCIA	*PRODUCTO*	*UNID.*	*PRECIO UNIT.*	*PRECIO*
HW-SPAR3AN-SK-UNI-G	Xilinx Spartan–3AN Starter Kit Board	1	239.01€	239.01€
EF-VIVADO-DESIGN-NL	Vivado Design Suite: Design Edition	1	2273.24€	2273.24€
DA-70146	Adapter USB 2.0 to serial M/DB-9	1	14.40€	14.40€
MG CHEMICALS 540	PCB COPPER CLAD 1/16"DBL SIDE	1	5.75€	5.75€
FX2-100P-1.27SVL(71)	CONN HDR 100POS 1.27MM	1	6.37€	6.37€
AD9752ARU	IC DAC 12bit 125MSPS HP 28-TSSOP	1	17.89€	17.89€
ADS805E	IC 12bit 20MHz A/D 28-SSOP	1	20.57€	20.57€
OPA354AIDBVR	IC OpAmp VFB 100MHz RRO SOT23-5	1	1.62€	1.62€
LTC6912CGN-1#PBF	IC OpAmp PGA 33MHZ RRO 16SSOP	1	3.67€	3.67€
A97594-ND	CONN SMA JACK STR 50 OHM PCB	2	1.99€	3,98€
—	SMD Resistors	10	0.10€	1.00€
—	SMD Capacitors	20	0.30€	6.00€
—	Varios (Pasta de soldar, hilo, ...)	—	—	35.00€
—	Mano de obra (Programación, soldadura, ...)	100	40€	4000€
			TOTAL:	**6628.5€**

Referencias

[1] *Spartan-3A/AN FPGA Starter Kit Board User Guide*. Xilinx, Inc. Versión 1.1, 2008. ∇ 11, 12, 23, 24, 27, 38, 42, 43, 44, 45, 48, 50, 52, 53, 54, 55, 56, 57, 58, 65, 73, 139

[2] *Download Qt Open Source*. The Qt Company. Último acceso: Agosto 1017. ∇ 12, 23, 76, 77

[3] *Free project scheduling and management app*. Gantt Project. Último acceso: Agosto 1017. ∇ 25

[4] *Inkscape*.El Equipo Inkscape. Último acceso: Agosto 1017. ∇ 27, 85

[5] *Spartan-3AN FPGA Board User's Manual*. Humandata, Ltd. Versión 1.1, 2009. ∇ 11, 39

[6] *Spartan-3AN FPGA Family Data Sheet*. Xilinx, Inc. Versión 4.2, 2014. ∇ 11, 40, 41

[7] *Spartan-3A/3AN Starter Kit Board Schematic*. Xilinx, Inc. Última Revisión: 2008. ∇ 11, 42

[8] *Configuring Xilinx FPGAs with SPI Serial Flash*. Xilinx, Inc. Versión 1.3, 2010. ∇ 11, 43

[9] *AT45DB161D – 16Mb 2.7volt Only Serial DataFlash®*. Atmel, Copr. Rev. 0807E?01/01, 2001. ∇ 43

[10] *M25P16 – Micron M25P16 Serial Flash Embedded Memory*. Micron Technology, Inc. Rev. I 4/15 EN, 2011. ∇ 43

[11] *Hitachi HD44780 LCD Controller Compatible*. 123Microcontroller.com. Último acceso: Agosto 1017. ∇ 11, 49

[12] *US ASCII, ANSI X3.4-1986 (ISO 646 International Reference Version)*. Columbia University. Última consulta: Agosto 2017. ∇ 50

[13] *HITACHI HD44780U (LCD-II)*. Hitachi, Ltd. Revisión 0.0, 1998. ∇ 52

[14] *Using Suspend Mode in Extended Spartan-3A Family FPGAs*. Marc Baker, Xilinx, Inc. Versión 1.1, 2014. ∇ 54

[15] *Scalable Vector Graphics*. Wikipedia. Último acceso: Agosto 2017. ∇ 85

[16] *KiCad Getting Started*. KiCad. Último acceso: Agosto 2017. ∇ 87, 139

[17] *Realterm: Serial Terminal*. Realterm. Último acceso: Agosto 2017. ∇ 87

[18] *Convertidor Digital → Analógico AD9752*. Analog Devices, Inc. Última actualización: 23/02/2017. ∇ 15, 107, 140

[19] *General LCD Testing Program*. Digilent®, Inc. Dan Pederson, 2004, Barron Barnett, 2004, Jacob Beck, 2006. ∇ 114

[20] *12-Bit, 20MHz Sampling ANALOG-TO-DIGITAL CONVERTER*. Texas Instruments. Enero de 1997. Últ. revisión: noviembre de 2002. ∇ 10, 15, 115, 141, 142

[21] *Dual Programmable Gain Amplifiers with Serial Digital Interface*. Linear Technology. 2004. ∇ 10, 15, 123, 141, 142

[22] *Qt Documentation → Qt Serial Port*. The Qt Company. Último acceso: Agosto 2017. ∇ 135

[23] *QCustomPlot©, an easy to use, modern plotting widget for Qt*. Emanuel Eichhammer. 2011–2015. ∇ 136

[24] *FX2 Series – 1.27mm Pitch Multi-function Two Piece Connectors*. Hirose Electric CO., LTD. Mayo de 2016. ∇ 10

[25] *OPAx354 250-MHz, Rail-to-Rail I/O, CMOS Operational Amplifiers*. Texas Instruments, Inc. 2016. ∇ 10, 140

[26] *MSO/DS1000Z Series Digital Oscilloscope*. RIGOL TECHNOLOGIES, Inc. Agosto de 2016. ∇ 15, 147

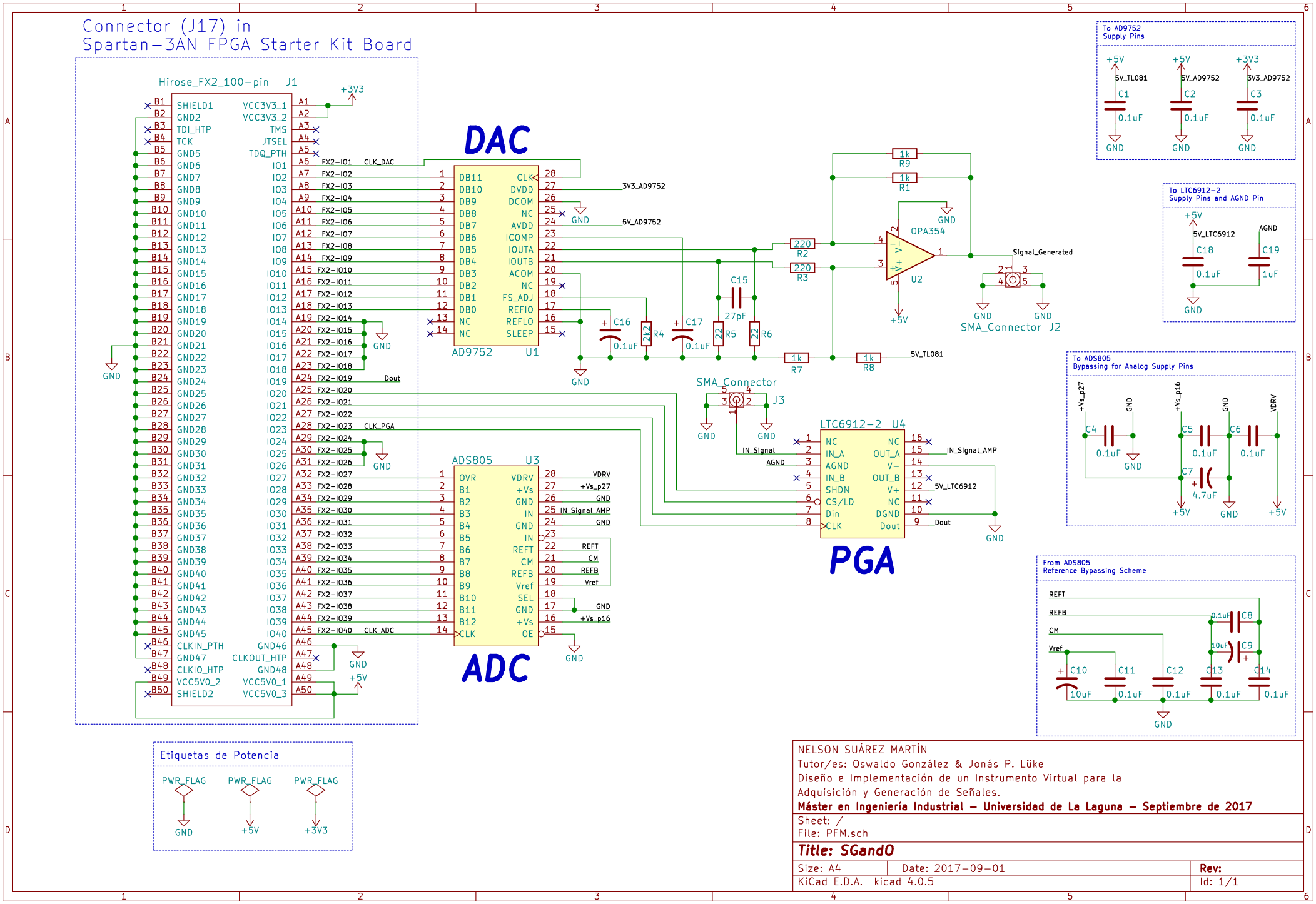
[27] *Generador de Funciones GF–232*. PROMAX Electrónica S.L. Versión: 01/97. ∇ 15, 147, 148

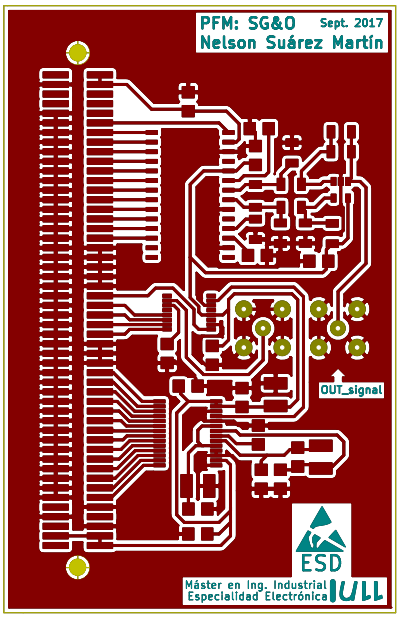
[28] *SMA Connector*. Radiall™. Última consulta: Agosto 2017. ∇ 10

Anexos

Anexo A

Esquemáticos


Connector (J17) in
Spartan-3AN FPGA Starter Kit Board
Hirose_FX2_100-pin J1
DAC
AD9752 U1
ADC
ADS805 U3
PGA
LTC6912-2 U4
OPA354 U2
SMA_Connector J2
SMA_Connector J3
To AD9752
Supply Pins
To LTC6912-2
Supply Pins and AGND Pin
To ADS805
Bypassing for Analog Supply Pins
From ADS805
Reference Bypassing Scheme
Etiquetas de Potencia
PWR_FLAG
NELSON SUÁREZ MARTÍN
Tutor/es: Oswaldo González & Jonás P. Lüke
Diseño e Implementación de un Instrumento Virtual para la
Adquisición y Generación de Señales.
Máster en Ingeniería Industrial – Universidad de La Laguna – Septiembre de 2017
Sheet: /
File: PFM.sch
Title: SGandO
Size: A4
Date: 2017-09-01
Rev:
KiCad E.D.A. kicad 4.0.5
Id: 1/1



NELSON SUÁREZ MARTÍN
Tutor/es: Oswaldo González & Jonás P. Lüke
Diseño e Implementación de un Instrumento Virtual para la Adquisición y Generación de Señales.
Máster en Ingeniería Industrial – Universidad de La Laguna – Septiembre de 2017

Sheet:
File: PFM.kicad_pcb

Title: SGandO

Size: A4	Date: 2017-09-01	**Rev: Escala 1:1**
KiCad E.D.A. kicad 4.0.5		Id: 1/1

Anexo B

Códigos VHDL

SGandO.vhd

```
------------------------------------------------------------------
-- NELSON SUÁREZ MARTÍN                                SGandO.vhd --
------------------------------------------------------------------
-- Máster en Ingeniería Industrial                               --
-- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
--          para la Adquisición y Generación de Señales.      --
-- Fecha: Septiembre 2017.                                       --
-- Tutor/es: Oswaldo González & Jonas Lüke                       --
-- Universidad de La Laguna                                      --
--                                                               --
------------------------------------------------------------------
--                          DESCRIPCIÓN                          --
------------------------------------------------------------------
-- Bloque general del proyecto donde se realiza el conexionado--
-- de los módulos específicos.                                   --
--                                                               --
------------------------------------------------------------------
```

```
18 --                      SGandO COMPONENTS                        --
19 -----------------------------------------------------------------
20 -- > RST_MODULE: Realiza un reset general al cargar el          --
21 --     programa, inicializando así todas las señales y          --
22 --     variables del mismo.                                     --
23 --                                                              --
24 -- > UART: Es el módulo encargado de la transmisión y           --
25 --     recepción de datos del puerto serie RS-232.              --
26 --                                                              --
27 -- > STATE_MACHINE: Módulo que actúa como intermediario         --
28 --     entre las órdenes enviadas por el software del PC y la   --
29 --     FPGA. Lleva a cabo también el control de envío de        --
30 --     datos hacia este software.                               --
31 --                                                              --
32 -- > SIGNAL_GENERATOR: Grupo de componentes encargados de       --
33 --     la preparación de datos para su envío hacia el DAC y     --
34 --     su conversión posteriormente en funciones. El envío de   --
35 --     parámetros se puede realizar desde la placa o el         --
36 --     software de PC.                                          --
37 --                                                              --
38 -- > OSCILLOSCOPE: Grupo de componentes encargados del          --
39 --     tratamiento de datos enviados por el ADC. Se encarga     --
40 --     también de la configuración del PGA.                     --
41 --                                                              --
42 -----------------------------------------------------------------
43 --                       SGandO INPUTS                           --
44 -----------------------------------------------------------------
45 -- > clk: Señal de reloj del sistema realizado (50MHz).         --
46 --                                                              --
47 -- > rst: Señal que realiza un reset común a todos los          --
48 --     módulos que componen 'SGandO'.                           --
```

```
49 --                                                                --
50 -- > ROT_A y ROT_B: Sentido de rotación del interruptor          --
51 --     rotativo de la placa.                                      --
52 --                                                                --
53 -- > selector_signal: Selección de la señal que se desea         --
54 --     generar.                                                   --
55 --                                                                --
56 -- > freq_mult_sw: Cambio de escala para las señales generadas--
57 --     cuando el control de 'signal_generator' lo tiene la       --
58 --     placa y no el software de PC.                              --
59 --                                                                --
60 -- > sel_param: Selección del parámetro que se quiere            --
61 --     modificar pulsando el interruptor rotativo cuando la      --
62 --     placa tiene el control de cambio de los mismos.            --
63 --                                                                --
64 -- > rx: Recepción de datos del puerto serie RS232.              --
65 --                                                                --
66 -- > data_ADC: Datos proporcionados por el ADC al módulo         --
67 --     'oscilloscope'.                                            --
68 --                                                                --
69 -- > OVR: El ADC indica a través de este bit si la señal está --
70 --     saturada.                                                  --
71 --                                                                --
72 -- > D_out: Reenvío que realiza el PGA de la ganancia que le  --
73 --     habíamos proporcionado en última instancia para           --
74 --     confirmar así su correcta recepción.                       --
75 --                                                                --
76 ------------------------------------------------------------------
77 --                      SGandO OUTPUTS                            --
78 ------------------------------------------------------------------
79 -- > tx: Transmisión de datos del puerto serie RS232.            --
```

```
80  --                                                                  --
81  -- > clk_DAC: Reloj del DAC.                                        --
82  --                                                                  --
83  -- > data_DAC: Envío de datos hacia el DAC.                         --
84  --                                                                  --
85  -- > clk_ADC: Reloj del ADC.                                        --
86  --                                                                  --
87  -- > OVR_led: Led indicador de la saturación del ADC.               --
88  --                                                                  --
89  -- > clk_PGA: Reloj del PGA.                                        --
90  --                                                                  --
91  -- > SHDN: Señal de reset del PGA.                                  --
92  --                                                                  --
93  -- > load_data: Habilitar/Deshabilitar el envío de una nueva        --
94  --     ganancia al PGA.                                             --
95  --                                                                  --
96  -- > D_in: Datos de entrada al PGA con la nueva ganancia.           --
97  --    - D_in<0-3> -> Ganancia Canal A.                              --
98  --    - D_in<4-7> -> Ganancia Canal B.                              --
99  --                                                                  --
100 -- > RS: Selector de registros del display LCD de la placa.         --
101 --                                                                  --
102 -- > RW: Control de lectura y escritura del display LCD de          --
103 --     la placa.                                                    --
104 --                                                                  --
105 -- > E: Habilitar/Deshabilitar la lectura y escritura del           --
106 --     display LCD de la placa.                                     --
107 --                                                                  --
108 -- > DB: Datos de entrada/salida del display LCD de la placa.       --
109 --                                                                  --
110 -- > ctrl_led: Indicador led (LED<7>) del control de                --
```

```
111 --     parámetros de 'signal_generator'.                        --
112 --    - '1' -> El control del generador de señales lo tiene     --
113 --             el software.                                     --
114 --    - '0' -> El control del generador de señales lo tiene     --
115 --             la placa.                                        --
116 --                                                              --
117 -- > sg_led: LED<4-1>.                                         --
118 --    - LED<4> -> '1' -> Generación de señal senoidal.          --
119 --    - LED<3> -> '1' -> Generación de señal triangular.        --
120 --    - LED<2> -> '1' -> Generación de señal cuadrada.          --
121 --    - LED<1> -> '1' -> Señal continua.                        --
122 --                                                              --
123 ----------------------------------------------------------------
124
125 library IEEE;
126 use IEEE.STD_LOGIC_1164.ALL;
127 use work.parametros.all;
128
129 entity SGandO is
130    port(  clk : in std_logic;
131           rst : in std_logic;
132 -- PARÁMETROS
133          ROT_A : in std_logic;
134          ROT_B : in std_logic;
135    sel_signal : in std_logic_vector(1 downto 0);
136  freq_mult_sw : in std_logic;
137     sel_param : in std_logic;
138 -- UART
139            rx : in std_logic;
140            tx : out std_logic;
141 -- DAC
```

```
142       clk_DAC : out std_logic;
143      data_DAC : out std_logic_vector(nbits-1 downto 0);
144  -- ADC
145      data_ADC : in std_logic_vector(nbits-1 downto 0);
146           OVR : in std_logic;
147       clk_ADC : out std_logic;
148       OVR_led : out std_logic;
149  -- PGA
150       clk_PGA : out std_logic;
151          SHDN : out std_logic;
152     load_data : out std_logic;
153          D_in : out std_logic;
154         D_out : in std_logic;
155  -- LCD
156            RS : out std_logic;
157            RW : out std_logic;
158             E : out std_logic;
159            DB : out std_logic_vector (nbits_com-1 downto 0);
160  -- LED<0-7>
161      ctrl_led : out std_logic;
162        sg_led : out std_logic_vector(3 downto 0));
163
164  end SGand0;
165
166  architecture Behavioral of SGand0 is
167
168  component rst_module is
169     port( clk : in std_logic;
170           rst : in std_logic;
171         reset : out std_logic);
172  end component rst_module;
```

```
173
174 -- SEÑALES DE RST_MODULE ---------------------------------------
175 signal reset : std_logic;
176 ------------------------------------------------------------------
177
178 component UART is
179    generic( baud_rate : natural;
180               paridad : boolean);
181    port(      clk : in std_logic;
182              reset : in std_logic;
183                 rx : in std_logic;
184                 tx : out std_logic;
185     transmit_dato : in std_logic;
186  dato_transmitido : out std_logic;
187           dato_in : in std_logic_vector(nbits_com-1 downto 0);
188     dato_recibido : out std_logic;
189        dato_leido : in std_logic;
190         error_chk : out std_logic;
191          dato_out : out std_logic_vector(nbits_com-1 downto 0));
192 end component UART;
193
194 -- SEÑALES DE LA UART -----------------------------------------
195 signal transmit_dato, dato_transmitido, dato_recibido,
    dato_leido, error_chk : std_logic;
196 ------------------------------------------------------------------
197
198 component state_machine is
199    port(    clk : in std_logic;
200           reset : in std_logic;
201   dato_recibido : in std_logic;
202   sel_signal_in : in std_logic_vector(1 downto 0);
```

```
203         data_in : in std_logic_vector(nbits_com-1 downto 0);
204        osc_data : in std_logic_vector(nbits_com-1 downto 0);
205      sg_enabled : out std_logic;
206      pc_control : out std_logic;
207     rst_ADC_clk : out std_logic;
208         rst_pga : out std_logic;
209      dato_leido : out std_logic;
210   transmit_dato : out std_logic;
211  sel_signal_out : out std_logic_vector(1 downto 0);
212            freq : out natural range 1 to 9;
213       freq_mult : out std_logic_vector(1 downto 0);
214               D : out integer range 0 to 8;
215           volts : out palabra;
216          offset : out palabra;
217        adq_time : out natural range 3 to adq_count;
218            gain : out std_logic_vector(3 downto 0);
219         trigger : out std_logic_vector(7 downto 0);
220    read_pointer : out integer range 0 to L_samples;
221       f_clk_DAC : out std_logic_vector(1 downto 0);
222        data_out : out std_logic_vector(nbits_com-1 downto 0));
223 end component state_machine;
224
225 -- SEÑALES DE LA MÁQUINA DE ESTADOS ----------------------------
226 signal rst_ADC_clk, rst_PGA, pc_control, sg_enabled : std_logic;
227 signal freq_sig : natural range 1 to 9;
228 signal D : integer range 0 to 8;
229 signal volts, offset : palabra;
230 signal data_in, osc_data, data_out, trigger: std_logic_vector(
      nbits_com-1 downto 0);
231 signal sel_signal_sig, freq_mult, f_clk_DAC : std_logic_vector(1
      downto 0);
```

```
232 signal read_pointer : integer range 0 to L_samples;
233 signal adq_time : natural range 3 to adq_count;
234 signal gain : std_logic_vector(3 downto 0);
235 ----------------------------------------------------------------
236
237 component signal_generator is
238    port(  clk : in std_logic;
239         reset : in std_logic;
240    sg_enabled : in std_logic;
241    pc_control : in std_logic;
242     f_clk_DAC : in std_logic_vector(1 downto 0);
243    sel_signal : in std_logic_vector(1 downto 0);
244     sel_param : in std_logic;
245  freq_mult_sw : in std_logic;
246     freq_mult : in std_logic_vector(1 downto 0);
247         ROT_A : in std_logic;
248         ROT_B : in std_logic;
249          freq : in natural range 1 to 9;
250             D : in integer range 0 to 8;
251      volts_pc : in palabra;
252     offset_pc : in palabra;
253        sg_led : out std_logic_vector(3 downto 0);
254      ctrl_led : out std_logic;
255            RS : out std_logic;
256            RW : out std_logic;
257             E : out std_logic;
258            DB : out std_logic_vector (nbits_com-1 downto 0);
259       clk_DAC : out std_logic;
260      data_DAC : out std_logic_vector(nbits-1 downto 0));
261 end component signal_generator;
262
```

```
263 component oscilloscope is
264    port(  clk : in std_logic;
265         reset : in std_logic;
266 -- ADQ_MODULE
267   rst_ADC_clk : in std_logic;
268      data_ADC : in std_logic_vector(nbits-1 downto 0);
269  read_pointer : in integer range 0 to L_samples;
270      adq_time : in natural range 3 to adq_count;
271       trigger : in std_logic_vector(7 downto 0);
272           OVR : in std_logic;
273       OVR_led : out std_logic;
274       clk_ADC : out std_logic;
275      osc_data : out std_logic_vector(nbits_com-1 downto 0);
276 -- PGA
277       rst_PGA : in std_logic;
278          gain : in std_logic_vector(3 downto 0);
279         D_out : in std_logic;
280          D_in : out std_logic;
281          SHDN : out std_logic;
282     load_data : out std_logic;
283       clk_PGA : out std_logic);
284 end component oscilloscope;
285
286 begin
287
288 rst_unit : rst_module
289    PORT MAP( clk => clk,
290              rst => rst,
291            reset => reset);
292
293 uart_unit : UART
```

```
294     GENERIC MAP(baud_rate => tasa_baudios,
295                 paridad => paridad_uart)
296     PORT MAP(  clk => clk,
297              reset => reset,
298                 rx => rx,
299                 tx => tx,
300      transmit_dato => transmit_dato,
301   dato_transmitido => dato_transmitido,
302            dato_in => data_in,
303      dato_recibido => dato_recibido,
304         dato_leido => dato_leido,
305          error_chk => error_chk,
306           dato_out => data_out);
307
308 state_machine_unit : state_machine
309     PORT MAP( clk => clk,
310             reset => reset,
311     dato_recibido => dato_recibido,
312     sel_signal_in => sel_signal,
313           data_in => data_out,
314          osc_data => osc_data,
315        sg_enabled => sg_enabled,
316        pc_control => pc_control,
317       rst_ADC_clk => rst_ADC_clk,
318           rst_PGA => rst_PGA,
319        dato_leido => dato_leido,
320     transmit_dato => transmit_dato,
321    sel_signal_out => sel_signal_sig,
322              freq => freq_sig,
323         freq_mult => freq_mult,
324                 D => D,
```

```
325             volts => volts,
326            offset => offset,
327          adq_time => adq_time,
328              gain => gain,
329           trigger => trigger,
330      read_pointer => read_pointer,
331         f_clk_DAC => f_clk_DAC,
332          data_out => data_in);
333
334 signal_generator_unit : signal_generator
335    PORT MAP( clk => clk,
336              reset => reset,
337        sg_enabled => sg_enabled,
338        pc_control => pc_control,
339         f_clk_DAC => f_clk_DAC,
340        sel_signal => sel_signal_sig,
341         sel_param => sel_param,
342      freq_mult_sw => freq_mult_sw,
343         freq_mult => freq_mult,
344             ROT_A => ROT_A,
345             ROT_B => ROT_B,
346              freq => freq_sig,
347                 D => D,
348          volts_pc => volts,
349         offset_pc => offset,
350            sg_led => sg_led,
351          ctrl_led => ctrl_led,
352                RS => RS,
353                RW => RW,
354                 E => E,
355                DB => DB,
```

```
356           clk_DAC => clk_DAC,
357          data_DAC => data_DAC);
358
359  oscilloscope_unit : oscilloscope
360     PORT MAP( clk => clk,
361             reset => reset,
362       rst_ADC_clk => rst_ADC_clk,
363          data_ADC => data_ADC,
364      read_pointer => read_pointer,
365          adq_time => adq_time,
366           trigger => trigger,
367               OVR => OVR,
368           OVR_led => OVR_led,
369           clk_ADC => clk_ADC,
370          osc_data => osc_data,
371           rst_PGA => rst_PGA,
372              gain => gain,
373            D_out => D_out,
374              D_in => D_in,
375              SHDN => SHDN,
376         load_data => load_data,
377           clk_PGA => clk_PGA);
378
379  end Behavioral;
```

rst_module.vhd

```
1  ----------------------------------------------------------------
2  -- NELSON SUÁREZ MARTÍN                                        --
```

```
3  ----------------------------------------------------------------
4  -- Máster en Ingeniería Industrial                             --
5  -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
6  --          para la Adquisición y Generación de Señales.      --
7  -- Fecha: Septiembre 2017.                                     --
8  -- Tutor/es: Oswaldo González & Jonas Lüke                     --
9  -- Universidad de La Laguna                                    --
10 --                                                            --
11 ----------------------------------------------------------------
12 --                        DESCRIPCIÓN                         --
13 ----------------------------------------------------------------
14 -- Si se presiona el pulsador de reset SOUTH(T15) de la placa --
15 -- se realiza un reset general de todo el sistema. Al dejar de--
16 -- presionar, este reset durará aprox. 0.5 milisegundos más.  --
17 --                                                            --
18 ----------------------------------------------------------------
19 --                    RST_MODULE INPUTS                       --
20 ----------------------------------------------------------------
21 -- > clk: Señal de reloj a 50MHz.                             --
22 --                                                            --
23 -- > rst: Entrada de reset desde el pulsador SOUTH(T15) de la --
24 --     placa.                                                 --
25 --                                                            --
26 ----------------------------------------------------------------
27 --                    RST_MODULE OUTPUTS                      --
28 ----------------------------------------------------------------
29 -- > reset: Reset de todos los múdulos del proyecto.          --
30 --                                                            --
31 ----------------------------------------------------------------
32
33 library IEEE;
```

```
34 use IEEE.STD_LOGIC_1164.ALL;
35
36 entity rst_module is
37    port( clk : in std_logic;
38          rst : in std_logic;
39         reset : out std_logic);
40 end rst_module;
41
42 architecture Behavioral of rst_module is
43
44 begin
45
46 process(clk, rst)
47    constant cuenta_max : natural := 2**15;
48    variable cuenta : integer range 0 to cuenta_max := 0;
49 begin
50    if rst = '1' then
51       reset <= '1';
52       cuenta := 0;
53    elsif clk'event and clk = '1' then
54       if cuenta = cuenta_max then
55          reset <= '0';
56       else
57          reset <= '1';
58          cuenta := cuenta+1;
59       end if;
60    end if;
61 end process;
62
63 end Behavioral;
```

state_machine.vhd

```
1  ----------------------------------------------------------------
2  -- NELSON SUÁREZ MARTÍN                    state_machine.vhd --
3  ----------------------------------------------------------------
4  -- Máster en Ingeniería Industrial                            --
5  -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
6  --          para la Adquisición y Generación de Señales.      --
7  -- Fecha: Septiembre 2017.                                     --
8  -- Tutor/es: Oswaldo González & Jonas Lüke                    --
9  -- Universidad de La Laguna                                   --
10 --                                                           --
11 ----------------------------------------------------------------
12 --                        DESCRIPCIÓN                        --
13 ----------------------------------------------------------------
14 -- La máquina de estados se encarga de traducir las órdenes   --
15 -- enviadas por el software a través de la UART hacia el      --
16 -- resto de módulos.                                          --
17 --                                                           --
18 -- Así mismo, otro proceso se encarga de enviar los datos     --
19 -- proporcionados por 'oscilloscope' a la UART de manera      --
20 -- sincronizada.                                             --
21 --                                                           --
22 -- La ejecución de estos dos procesos se realiza mediante la  --
23 -- creación de un reloj cuya frecuencia es aprox. de 1.5 KHz. --
24 --                                                           --
25 ----------------------------------------------------------------
26 --                  state_machine INPUTS                     --
```

```
27 ----------------------------------------------------------------
28 -- > clk: Señal de reloj de la placa a 50 MHz.                 --
29 --                                                             --
30 -- > reset: Señal de reseteo del módulo.                       --
31 --                                                             --
32 -- > dato_recibido: Señal procedente del módulo "UART"         --
33 --     indicando que se ha recibido un nuevo dato.             --
34 --                                                             --
35 -- > sel_signal_in: 2 bits que indican al generador de         --
36 --     señales la forma de la misma:                           --
37 --    - '00': Generar señal senoidal.                          --
38 --    - '01': Generar señal triangular.                        --
39 --    - '11': Generar señal cuadrada.                          --
40 --    - '10': Señal continua.                                  --
41 --                                                             --
42 -- > data_in: 8 bits procedentes del módulo "UART" recibidos   --
43 --     desde el software.                                      --
44 --                                                             --
45 -- > osc_data: Datos de 8 bits procedentes del módulo          --
46 --     "oscilloscope".                                         --
47 --                                                             --
48 ----------------------------------------------------------------
49 --                  state_machine OUTPUTS                      --
50 ----------------------------------------------------------------
51 -- > sg_enabled: Habilita/Deshabilita el generador de señales.--
52 --                                                             --
53 -- > pc_control: Indica si los parámetros son modificados     --
54 --     desde el software o desde la placa.                     --
55 --    - '0': Los parámetros son modificados desde los          --
56 --     actuadores de la placa.                                 --
57 --    - '1': Los parámetros son modificados desde los          --
```

```
58 --     controles del software.                                    --
59 --                                                                --
60 -- > rst_ADC_clk: Reset de la señal de reloj del ADC cuando      --
61 --     existe un cambio de muestreo.                              --
62 --                                                                --
63 -- > rst_PGA: Reset del PGA cuando existe un cambio de           --
64 --     ganancia.                                                  --
65 --                                                                --
66 -- > dato_leido: Indica a la "UART" que ya se ha leído el        --
67 --     dato 'data_in'.                                            --
68 --                                                                --
69 -- > transmit_dato: Indica a la "UART" que puede transmitir      --
70 --     un nuevo dato.                                             --
71 --                                                                --
72 -- > sel_signal_out: Forma de la señal que se genera.           --
73 --                                                                --
74 -- > freq: El software indica los saltos que habrán en las       --
75 --     tablas de señales para su generación a la frecuencia      --
76 --     especificada en el software.                               --
77 --                                                                --
78 -- > freq_mult: Multiplicador de frecuencia que indica el        --
79 --     software.                                                  --
80 --                                                                --
81 -- > D: El software indica el ciclo de trabajo de la señal       --
82 --     a generar.                                                 --
83 --                                                                --
84 -- > volts: El software indica la tensión pico a pico de la      --
85 --     señal a generar.                                           --
86 --                                                                --
87 -- > offset: El software indica el offset de la señal a          --
88 --     generar.                                                   --
```

```
89  --                                                                --
90  -- > adq_time: El software indica el tiempo de adquisición    --
91  --     del ADC.                                                   --
92  --                                                                --
93  -- > gain: Ganancia que se le proporciona al PGA.             --
94  --                                                                --
95  -- > trigger: El software indica el nivel del umbral de       --
96  --     disparo para la visualización de la señal en la        --
97  --     pantalla del oscilospio del software.                  --
98  --                                                                --
99  -- > read_pointer: Marca el ritmo de envío de datos del       --
100 --     vector de entrada 'osc_data'.                              --
101 --                                                                --
102 -- > f_clk_DAC: Marca las diferentes frecuencias a las que el --
103 --     DAC muestrea.                                              --
104 --                                                                --
105 -- > data_out: Dato de 8 bits que se envía al módulo "UART"   --
106 --     para su posterior envío hacia el software.             --
107 --                                                                --
108 ----------------------------------------------------------------------
109
110 library IEEE;
111 use IEEE.STD_LOGIC_1164.ALL;
112 use IEEE.NUMERIC_STD.ALL;
113 use work.parametros.all;
114
115 entity state_machine is
116    port(    clk : in std_logic;
117           reset : in std_logic;
118   dato_recibido : in std_logic;
119   sel_signal_in : in std_logic_vector(1 downto 0);
```

```
120         data_in : in std_logic_vector(nbits_com-1 downto 0);
121        osc_data : in std_logic_vector(nbits_com-1 downto 0);
122      sg_enabled : out std_logic;
123      pc_control : out std_logic;
124     rst_ADC_clk : out std_logic;
125         rst_PGA : out std_logic;
126      dato_leido : out std_logic;
127   transmit_dato : out std_logic;
128  sel_signal_out : out std_logic_vector(1 downto 0);
129            freq : out natural range 1 to 9;
130       freq_mult : out std_logic_vector(1 downto 0);
131               D : out integer range 0 to 8;
132           volts : out palabra;
133          offset : out palabra;
134        adq_time : out natural range 3 to adq_count;
135            gain : out std_logic_vector(3 downto 0);
136         trigger : out std_logic_vector(7 downto 0);
137    read_pointer : out integer range 0 to L_samples;
138       f_clk_DAC : out std_logic_vector(1 downto 0);
139        data_out : out std_logic_vector(nbits_com-1 downto 0));
140 end state_machine;
141
142 architecture Behavioral of state_machine is
143
144 signal reloj, send_data : std_logic := '0';
145 signal pc_control_sig : std_logic := '1';
146 signal sel_signal_sig : std_logic_vector(1 downto 0) := (others
    => '0');
147 signal offset_sig, volts_sig : std_logic_vector(nbits-1 downto
    0) := (others => '0');
```

```
148 signal adq_times : std_logic_vector(4 downto 0) := (others =>
        '0');
149
150 begin
151
152 -- Reloj de la máquina de estados a 1.5KHz aprox.
153 clk_maq_estados : process(clk, reset)
154    constant cuenta_max : natural := 2**15;
155    variable cuenta : integer range 0 to cuenta_max := 0;
156 begin
157    if reset = '1' then
158       cuenta := 0;
159       reloj <= '1';
160    elsif clk'event and clk = '1' then
161       cuenta := (cuenta + 1) mod cuenta_max;
162       if cuenta < cuenta_max/2 then
163          reloj <= '1';
164       else
165          reloj <= '0';
166       end if;
167    end if;
168 end process;
169
170 -- Tratamiento de las órdenes recibidas desde la UART que fueron
171 -- enviadas por el PC.
172 maq_estados : process(reloj, reset)
173    variable state : states := WAITING;
174    variable freq_int : natural range 1 to 31 := 1;
175    variable strange : std_logic := '0';
176 begin
177    if reset = '1' then
```

```
178         state := RESTART;
179         sg_enabled <= '1';
180         pc_control_sig <= '1'; -- Controlado desde un principio
               desde el software.
181         sel_signal_sig <= (others => '0');
182         freq <= 1;
183         freq_int := 1;
184         freq_mult <= (others => '0');
185         f_clk_DAC <= (others => '0');
186         D <= 4;
187         volts_sig <= "000010000000";
188         offset_sig <= (others => '0');
189         trigger <= (others => '0');
190         adq_times <= (others => '0');
191         dato_leido <= '0';
192         rst_ADC_clk <= '0';
193         rst_PGA <= '0';
194         gain <= "0001";
195         strange := '0';
196      elsif reloj'event and reloj = '1' then
197         rst_ADC_clk <= '0';
198         rst_PGA <= '0';
199         case state is
200            when WAITING =>
201               if dato_recibido = '1' then
202                  dato_leido <= '1';
203                  case data_in is
204                     when "00000000" => -- Generador de señales
                           fuera de servicio.
205                        sg_enabled <= '0';
```

```
206                     when "00011111" => -- Generador de señales
                           activado.
207                        sg_enabled <= '1';
208                     when "00000011" => -- El generador de señales
                           es controlado desde el PC.
209                        pc_control_sig <= '1';
210                        state := RESTART;
211                     when "00000010" => -- El generador de señales
                           es controlado desde la placa.
212                        pc_control_sig <= '0';
213                        state := RESTART;
214                     when "00000001" => -- Reset de la máquina de
                           estados.
215                        state := RESTART;
216                     when "00001010" => -- Señal senoidal.
217                        sel_signal_sig <= "00";
218                     when "00001011" => -- Señal triangular.
219                        sel_signal_sig <= "01";
220                     when "00001100" => -- Señal cuadrada.
221                        sel_signal_sig <= "11";
222                     when "00001101" => -- Señal contínua.
223                        sel_signal_sig <= "10";
224                     when "00000100" => -- Cambio de escala de
                           frecuencia x1.
225                        f_clk_DAC <= "00";
226                     when "00000101" => -- Cambio de escala de
                           frecuencia x100.
227                        f_clk_DAC <= "01";
228                     when "00000110" => -- Solo para generar
                           señales con frecuencia = 1 MHz.
229                        strange := '1';
```

```
230                         f_clk_DAC <= "10";
231                         freq <= 3;
232                         freq_mult <= "11";
233                      when others =>
234                         case data_in(7 downto 5) is
235                            when "001" =>
236                               if strange = '1' then
237                                  f_clk_DAC <= "01";
238                                  strange := '0';
239                               end if;
240                               freq_int := to_integer(unsigned(
                                     data_in(4 downto 0)));
241                               if freq_int > 18 then
242                                  freq <= freq_int - 18;
243                                  freq_mult <= "11";
244                               elsif freq_int > 9 then
245                                  freq <= freq_int - 9;
246                                  freq_mult <= "10";
247                               else
248                                  freq <= freq_int;
249                                  freq_mult <= "00";
250                               end if;
251                            when "010" =>
252                               volts_sig <= (data_in(4 downto 0))
                                     &"0000000";
253                            when "011" =>
254                               offset_sig <= (data_in(4 downto 0))
                                     &"0000000";
255                            when "100" =>
256                               D <= to_integer(unsigned(data_in(4
                                     downto 0)));
```

```
257                         when "101" =>
258                            adq_times <= data_in(4 downto 0);
259                            rst_ADC_clk <= '1';
260                         when "110" =>
261                            trigger <= data_in(4 downto 0)&"000";
262                         when "111" =>
263                            gain <= data_in(3 downto 0);
264                            rst_PGA <= '1';
265                         when others =>
266                            state := WAITING;
267                      end case;
268                end case;
269             else
270                dato_leido <= '0';
271                state := WAITING;
272             end if;
273          when RESTART =>
274             state := WAITING;
275             sg_enabled <= '1';
276             sel_signal_sig <= (others => '0');
277             freq <= 1;
278             freq_int := 1;
279             freq_mult <= (others => '0');
280             f_clk_DAC <= (others => '0');
281             D <= 4;
282             volts_sig <= "000010000000";
283             offset_sig <= (others => '0');
284             trigger <= (others => '0');
285             adq_times <= (others => '0');
286             dato_leido <= '0';
287             rst_ADC_clk <= '0';
```

```
288             rst_PGA <= '0';
289             gain <= "0001";
290             strange := '0';
291          when others =>
292             state := WAITING;
293       end case;
294    end if;
295
296 end process;
297
298 pc_control <= pc_control_sig;
299 sel_signal_out <= sel_signal_sig when pc_control_sig = '1' else
      sel_signal_in;
300 offset <= palabra(offset_sig);
301 volts <= palabra(volts_sig);
302
303 with adq_times select
304    adq_time <= 2**16 when "00000",
305                2**15 when "00001",
306                2**14 when "00010",
307                2**13 when "00011",
308                2**12 when "00100",
309                2**11 when "00101",
310                2**10 when "00110",
311             (2**10)/2 when "00111",
312                  2**9 when "01000",
313                   384 when "01001",
314                  2**8 when "01010",
315                  2**7 when "01011",
316                  2**6 when "01100",
317                  2**5 when "01101",
```

```
318                  2**4 when "01110",
319                    15 when "01111",
320                    10 when "10000",
321                     5 when "10001",
322                  2**2 when "10010",
323                     3 when others;
324
325 -- Proceso de adquisición de datos desde 'oscilloscope' hacia
326 -- la UART para su envío al PC.
327 osc_unit : process(reloj, reset)
328    variable read_point : integer range 0 to L_samples := 0;
329 begin
330    if reset = '1' then
331       send_data <= '0';
332       read_point := 0;
333       read_pointer <= 0;
334    elsif reloj'event and reloj = '0' then
335       if read_point < L_samples then
336          send_data <= '1';
337          data_out <= osc_data;
338          read_point := read_point+1;
339          read_pointer <= read_point;
340       else
341          read_point := 0;
342          read_pointer <= read_point;
343          send_data <= '0';
344       end if;
345    end if;
346 end process;
347
348 transmit_dato <= send_data and reloj;
```

```
349
350 end Behavioral;
```

signal_generator.vhd

```
1  ----------------------------------------------------------------
2  -- NELSON SUÁREZ MARTÍN                  signal_generator.vhd --
3  ----------------------------------------------------------------
4  -- Máster en Ingeniería Industrial                            --
5  -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
6  --          para la Adquisición y Generación de Señales.      --
7  -- Fecha: Septiembre 2017.                                     --
8  -- Tutor/es: Oswaldo González & Jonas Lüke                    --
9  -- Universidad de La Laguna                                   --
10 --                                                           --
11 ----------------------------------------------------------------
12 --                        DESCRIPCIÓN                        --
13 ----------------------------------------------------------------
14 -- El generador de señales es el encargado de leer de las     --
15 -- tablas creadas en 'real2bit.vhd' los valores para cada una --
16 -- de las señales y enviarlos al módulo del DAC.              --
17 --                                                           --
18 ----------------------------------------------------------------
19 --               signal_generator COMPONENTS                 --
20 ----------------------------------------------------------------
21 -- > PARAM_SG: Describe cómo va a ser la señal y qué          --
22 --     características tendrá, y si los parámetros a enviar   --
23 --     son los que vienen de la placa o los enviados por el   --
24 --     software.                                             --
25 --                                                           --
```

```
26 -- > DAC: Encargado de tomar los datos que le proporciona     --
27 --     el módulo actual (signal_generator.vhd) y enviarlos al --
28 --     vector de salida hacia el DAC físico.                  --
29 --                                                            --
30 -- > LCD: Máquina de estados para visualizar los parametros   --
31 --     que estamos cambiando cuando el generador de señales es--
32 --     controlado desde la placa.                             --
33 --                                                            --
34 ----------------------------------------------------------------
35 --                 signal_generator INPUTS                    --
36 ----------------------------------------------------------------
37 -- > clk: Señal de reloj de la placa a 50 MHz.                --
38 --                                                            --
39 -- > reset: Señal de reseteo del módulo.                      --
40 --                                                            --
41 -- > sg_enabled: Habilita/Deshabilita el generador de señales.--
42 --                                                            --
43 -- > pc_control: Indica si los parámetros son modificados     --
44 --     desde el software o desde la placa.                    --
45 --    - '0': Los parámetros son modificados desde los         --
46 --     actuadores de la placa.                                --
47 --    - '1': Los parámetros son modificados desde los         --
48 --     controles del software.                                --
49 --                                                            --
50 -- > f_clk_DAC: Marca las diferentes frecuencias a las que el --
51 --     DAC muestrea.                                          --
52 --                                                            --
53 -- > sel_signal: Forma de la señal que se genera.             --
54 --                                                            --
55 -- > sel_param: Selección del parámetro que se quiere         --
56 --     modificar pulsando el interruptor rotativo cuando la   --
```

```
57 --     placa tiene el control de cambio de los mismos.        --
58 --                                                            --
59 -- > freq_mult_sw: Cambio de escala para las señales generadas--
60 --     cuando el control de 'signal_generator' lo tiene la    --
61 --     placa y no el software de PC.                          --
62 --                                                            --
63 -- > freq_mult: Multiplicador de frecuencia que indica el     --
64 --     software.                                              --
65 --                                                            --
66 -- > ROT_A y ROT_B: Sentido de rotación del interruptor       --
67 --     rotativo de la placa.                                  --
68 --                                                            --
69 -- > freq: El software indica los saltos que habrán en las    --
70 --     tablas de señales para su generación a la frecuencia   --
71 --     especificada en el software.                           --
72 --                                                            --
73 -- > D: El software indica el ciclo de trabajo de la señal    --
74 --     a generar.                                             --
75 --                                                            --
76 -- > volts_PC: El software indica la tensión pico a pico de la--
77 --     señal a generar.                                       --
78 --                                                            --
79 -- > offset_PC: El software indica el offset de la señal a    --
80 --     generar.                                               --
81 --                                                            --
82 ----------------------------------------------------------------
83 --                signal_generator OUTPUTS                    --
84 ----------------------------------------------------------------
85 -- > sg_led: LED<4-1>.                                        --
86 --    - LED<4> -> '1' -> Generación de señal senoidal.        --
87 --    - LED<3> -> '1' -> Generación de señal triangular.      --
```

```
88  --    - LED<2> -> '1' -> Generación de señal cuadrada.        --
89  --    - LED<1> -> '1' -> Señal continua.                      --
90  --                                                            --
91  -- > ctrl_led: Indicador led (LED<7>) del control de         --
92  --     parámetros de 'signal_generator'.                      --
93  --    - '1' -> El control del generador de señales lo tiene   --
94  --             el software.                                   --
95  --    - '0' -> El control del generador de señales lo tiene   --
96  --             la placa.                                      --
97  --                                                            --
98  -- > RS: Selector de registros del display LCD de la placa.   --
99  --                                                            --
100 -- > RW: Control de lectura y escritura del display LCD de    --
101 --     la placa.                                              --
102 --                                                            --
103 -- > E: Habilitar/Deshabilitar la lectura y escritura del     --
104 --     display LCD de la placa.                               --
105 --                                                            --
106 -- > DB: Datos de entrada/salida del display LCD de la placa. --
107 --                                                            --
108 -- > clk_DAC: Reloj del DAC.                                  --
109 --                                                            --
110 -- > data_DAC: Envío de datos hacia el DAC.                   --
111 --                                                            --
112 ----------------------------------------------------------------
113
114 library IEEE;
115 use IEEE.STD_LOGIC_1164.ALL;
116 use IEEE.STD_LOGIC_ARITH.ALL;
117 use IEEE.STD_LOGIC_UNSIGNED.ALL;
118 use work.parametros.all;
```

```
119 use work.real2bit.all;
120
121 entity signal_generator is
122 port(     clk : in std_logic;
123         reset : in std_logic;
124    sg_enabled : in std_logic;
125    pc_control : in std_logic;
126     f_clk_DAC : in std_logic_vector(1 downto 0);
127    sel_signal : in std_logic_vector(1 downto 0);
128     sel_param : in std_logic;
129  freq_mult_sw : in std_logic;
130     freq_mult : in std_logic_vector(1 downto 0);
131         ROT_A : in std_logic;
132         ROT_B : in std_logic;
133          freq : in natural range 1 to 9;
134             D : in integer range 0 to 8;
135      volts_pc : in palabra;
136     offset_pc : in palabra;
137        sg_led : out std_logic_vector(3 downto 0);
138      ctrl_led : out std_logic;
139            RS : out std_logic;
140            RW : out std_logic;
141             E : out std_logic;
142            DB : out std_logic_vector (nbits_com-1 downto 0);
143       clk_DAC : out std_logic;
144      data_DAC : out std_logic_vector(nbits-1 downto 0));
145 end signal_generator;
146
147 architecture Behavioral of signal_generator is
148
149 component DAC is
```

```
150     port( clk : in std_logic;
151         reset : in std_logic;
152     f_clk_DAC : in std_logic_vector(1 downto 0);
153       data_in : in std_logic_vector(nbits-1 downto 0);
154       clk_DAC : out std_logic;
155      data_out : out std_logic_vector(nbits-1 downto 0));
156 end component DAC;
157 
158 component param_SG is
159     port(  clk : in std_logic;
160          reset : in std_logic;
161     pc_control : in std_logic;
162  freq_mult_sw : in std_logic;
163      freq_mult : in std_logic_vector(1 downto 0);
164      f_clk_DAC : in std_logic_vector(1 downto 0);
165      sel_param : in std_logic;
166          ROT_A : in std_logic;
167          ROT_B : in std_logic;
168        freq_in : in natural range 1 to 9;
169              D : in integer range 0 to 8;
170       volts_pc : in palabra;
171      offset_pc : in palabra;
172       position : in integer range 0 to M-1;
173       freq_out : out natural range 1 to M-1;
174          ufreq : out std_logic_vector(1 downto 0);
175          volts : out palabra;
176         offset : out palabra;
177       freq_lcd : out std_logic_vector(nbits_lcd-1 downto 0);
178     escala_lcd : out std_logic_vector(1 downto 0);
179          D_lcd : out std_logic_vector(nbits_lcd-1 downto 0);
180      volts_lcd : out std_logic_vector(nbits_lcd-1 downto 0);
```

```
181     offset_lcd : out std_logic_vector(nbits_lcd downto 0);
182      param_lcd : out std_logic_vector(nbits_lcd-1 downto 0);
183       ctrl_led : out std_logic);
184 end component param_SG;
185 
186 -- SEÑALES DEL MÓDULO DE PARÁMETROS
    ----------------------------------------------------
187 signal freq_lcd, D_lcd, volts_lcd, param_lcd : std_logic_vector(
    nbits_lcd-1 downto 0);
188 signal offset_lcd : std_logic_vector(nbits_lcd downto 0);
189 signal f_clk_DAC_sig : std_logic_vector(1 downto 0);
190 ------------------------------------------------------------------------------
191 
192 component LCD is
193     port( clk : in std_logic;
194         reset : in std_logic;
195      sg_board : in std_logic;
196        escala : in std_logic_vector(1 downto 0);
197         param : in std_logic_vector(nbits_lcd-1 downto 0);
198          freq : in std_logic_vector(nbits_lcd-1 downto 0);
199             D : in std_logic_vector(nbits_lcd-1 downto 0);
200         volts : in std_logic_vector(nbits_lcd-1 downto 0);
201        offset : in std_logic_vector(nbits_lcd downto 0);
202            DB : out std_logic_vector(nbits_com-1 downto 0);
203            RS : out std_logic;
204            RW : out std_logic;
205             E : out std_logic);
206 end component LCD;
207 
208 signal clk_DAC_sig : std_logic := '0';
```

```
209 signal pointer, position : integer range 0 to M-1 := M-1;
210 signal freq_sig : natural range 1 to M-1 := 1;
211 signal dato, dato_previo : palabra := (others => '0');
212 signal volts, offset : palabra;
213 signal envio_dato : std_logic_vector(nbits-1 downto 0) := (
    others => '0');
214 signal escala_lcd : std_logic_vector(1 downto 0) := (others =>
    '0');
215
216 begin
217
218 DAC_unit : DAC
219    PORT MAP(clk => clk,
220             reset => reset,
221          f_clk_DAC => f_clk_DAC_sig,
222           data_in => envio_dato,
223           clk_DAC => clk_DAC_sig,
224          data_out => data_DAC);
225
226 clk_DAC <= clk_DAC_sig;
227
228 param_SG_unit : param_SG
229    PORT MAP(clk => clk,
230             reset => reset,
231        pc_control => pc_control,
232    freq_mult_sw => freq_mult_sw,
233          freq_mult => freq_mult,
234          f_clk_DAC => f_clk_DAC,
235          sel_param => sel_param,
236              ROT_A => ROT_A,
237              ROT_B => ROT_B,
```

```
238          freq_in => freq,
239               D => D,
240         volts_pc => volts_pc,
241        offset_pc => offset_pc,
242         position => position,
243         freq_out => freq_sig,
244            ufreq => f_clk_DAC_sig,
245            volts => volts,
246           offset => offset,
247         freq_lcd => freq_lcd,
248       escala_lcd => escala_lcd,
249            D_lcd => D_lcd,
250        volts_lcd => volts_lcd,
251       offset_lcd => offset_lcd,
252        param_lcd => param_lcd,
253         ctrl_led => ctrl_led);
254
255 LCD_unit : LCD
256    PORT MAP(CLK => clk,
257           reset => reset,
258         sg_board => pc_control,
259           escala => escala_lcd,
260            param => param_lcd,
261             freq => freq_lcd,
262                D => D_lcd,
263            volts => volts_lcd,
264           offset => offset_lcd,
265               DB => DB,
266               RS => RS,
267               RW => RW,
268                E => E);
```

```
269
270 -- Toma de datos para la generación de la señal.
271 GEN_SIGNAL : process(clk_DAC_sig, reset)
272 begin
273    if reset = '1' then
274       pointer <= M-1;
275    elsif clk_DAC_sig'event and clk_DAC_sig = '1' then
276       pointer <= (pointer + freq_sig) mod M;
277    end if;
278 position <= pointer;
279 end process GEN_SIGNAL;
280
281 -- Preparación y envío del dato hacia el DAC.
282 dato_previo <= tabla_senoidal(pointer) when sel_signal = "00"
       else
283              tabla_triangular(pointer) when sel_signal = "01"
                    else
284                tabla_cuadrada(pointer) when sel_signal = "11"
                       else
285                truncar(1.0);
286
287 dato <= extraer(volts*extraer(truncar(1.0)*dato_previo)) +
       offset;
288 envio_dato <= (dato + conv_signed(2**(nbits-1),nbits)) when
       sg_enabled = '1' else (others => '0');
289
290 -- Encendido del indicador LED correspondiente.
291 sg_led <= "1000" when sel_signal = "00" else
292           "0100" when sel_signal = "01" else
293           "0010" when sel_signal = "11" else
294           "0001";
```

```
295
296 end Behavioral;
```

DAC.vhd

```
1  ----------------------------------------------------------------
2  -- NELSON SUÁREZ MARTÍN                                DAC.vhd --
3  ----------------------------------------------------------------
4  -- Máster en Ingeniería Industrial                             --
5  -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
6  --          para la Adquisición y Generación de Señales.      --
7  -- Fecha: Septiembre 2017.                                     --
8  -- Tutor/es: Oswaldo González & Jonas Lüke                    --
9  -- Universidad de La Laguna                                   --
10 --                                                           --
11 ----------------------------------------------------------------
12 --                        DESCRIPCIÓN                        --
13 ----------------------------------------------------------------
14 -- Módulo que se encarga del envío de datos hacia el DAC      --
15 -- físico. Para ello, según la frecuencia requerida, generará --
16 -- también un reloj determinado.                              --
17 --                                                           --
18 ----------------------------------------------------------------
19 --                         DAC INPUTS                        --
20 ----------------------------------------------------------------
21 -- > clk: Señal de reloj de la placa a 50 MHz.               --
22 --                                                           --
23 -- > reset: Señal de reseteo del módulo.                     --
24 --                                                           --
25 -- > f_clk_DAC: Marca las diferentes frecuencias a las que el --
```

```
26 --     DAC muestrea.                                                --
27 --                                                                  --
28 -- > data_in: Datos proporcionados por el módulo principal         --
29 --     'signal_generator.vhd'.                                      --
30 --                                                                  --
31 ------------------------------------------------------------------
32 --                          DAC OUTPUTS                             --
33 ------------------------------------------------------------------
34 -- > clk_DAC: Reloj del DAC.                                        --
35 --                                                                  --
36 -- > data_out: Envío de datos hacia el DAC.                        --
37 --                                                                  --
38 ------------------------------------------------------------------
39
40 library IEEE;
41 use IEEE.STD_LOGIC_1164.ALL;
42 use work.parametros.all;
43
44 entity DAC is
45    port( clk : in std_logic;
46        reset : in std_logic;
47    f_clk_DAC : in std_logic_vector(1 downto 0);
48      data_in : in std_logic_vector(nbits-1 downto 0);
49      clk_DAC : out std_logic;
50     data_out : out std_logic_vector(nbits-1 downto 0));
51 end DAC;
52
53 architecture Behavioral of DAC is
54
55 signal reloj, reloj_a, reloj_b : std_logic := '0';
56
```

```
57 begin
58
59 -- Reloj a f = 400 KHz cuando la escala del generador de señales
60 -- es x1.
61 CLK_ENVIOS_DAC_A : process(clk, reset)
62    variable cuenta : integer range 0 to 64 := 0;
63 begin
64    if reset = '1' then
65       reloj_a <= '0';
66       cuenta := 0;
67    elsif clk'event and clk = '1' then
68       if cuenta = 0 then
69          reloj_a <= not reloj_a;
70       end if;
71       cuenta := (cuenta+1) mod 64;
72    end if;
73 end process CLK_ENVIOS_DAC_A;
74
75 -- Reloj a f = 25 MHz cuando queremos generar una señal a una
76 -- f = 1 MHz.
77 CLK_ENVIOS_DAC_B : process(clk, reset)
78 begin
79    if reset = '1' then
80       reloj_b <= '0';
81    elsif clk'event and clk = '1' then
82       reloj_b <= not reloj_b;
83    end if;
84 end process CLK_ENVIOS_DAC_B;
85
86 -- Cuando la escala del generador de señales es x100, el reloj
   del DAC es
```

```
87  -- directamente el reloj de 50 MHz de la placa de Xilinx.
88  reloj <= not clk when f_clk_DAC = "01" else reloj_a when
       f_clk_DAC = "00" else reloj_b;
89
90  -- Envío de datos hacia el DAC.
91  ENVIOS_DAC : process(reloj, reset)
92  begin
93     if reset = '1' then
94        data_out <= (others => '0');
95     elsif reloj'event and reloj = '1' then
96        data_out <= data_in;
97     end if;
98  end process;
99
100 clk_DAC <= reloj;
101
102 end Behavioral;
```

param_SG.vhd

```
1   ----------------------------------------------------------------
2   -- NELSON SUÁREZ MARTÍN                          param_SG.vhd --
3   ----------------------------------------------------------------
4   -- Máster en Ingeniería Industrial                           --
5   -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
6   --          para la Adquisición y Generación de Señales.     --
7   -- Fecha: Septiembre 2017.                                    --
8   -- Tutor/es: Oswaldo González & Jonas Lüke                   --
9   -- Universidad de La Laguna                                  --
10  --                                                           --
```

```
11 ----------------------------------------------------------------
12 --                         DESCRIPCIÓN                        --
13 ----------------------------------------------------------------
14 -- Realiza la parte más importante y delicada del generador   --
15 -- señales. Toma valores del software o de la placa según     --
16 -- órdenes del primero y genera los datos para la señal de    --
17 -- salida hacia el DAC.                                       --
18 --                                                            --
19 ----------------------------------------------------------------
20 --                     param_SG COMPONENTS                    --
21 ----------------------------------------------------------------
22 -- > ROTARY_SWITCH: Funcionamiento del interruptor rotativo   --
23 --     de la placa (ROT-A y ROT-B).                           --
24 --                                                            --
25 ----------------------------------------------------------------
26 --                       param_SG INPUTS                      --
27 ----------------------------------------------------------------
28 -- > clk: Señal de reloj de la placa a 50 MHz.                --
29 --                                                            --
30 -- > reset: Señal de reseteo del módulo.                      --
31 --                                                            --
32 -- > pc_control: Indica si los parámetros son modificados     --
33 --     desde el software o desde la placa.                    --
34 --    - '0': Los parámetros son modificados desde los         --
35 --     actuadores de la placa.                                --
36 --    - '1': Los parámetros son modificados desde los         --
37 --     controles del software.                                --
38 --                                                            --
39 -- > freq_mult_sw: Cambio de escala para las señales generadas--
40 --     cuando el control de 'signal_generator' lo tiene la    --
41 --     placa y no el software de PC.                          --
```

```
42 --                                                                --
43 -- > freq_mult: Multiplicador de frecuencia que indica el     --
44 --     software.                                                  --
45 --                                                                --
46 -- > f_clk_DAC: Marca las diferentes frecuencias a las que el --
47 --     DAC muestrea.                                              --
48 --                                                                --
49 -- > sel_param: Selección del parámetro que se quiere          --
50 --     modificar pulsando el interruptor rotativo cuando la   --
51 --     placa tiene el control de cambio de los mismos.        --
52 --                                                                --
53 -- > ROT_A y ROT_B: Sentido de rotación del interruptor       --
54 --     rotativo de la placa.                                      --
55 --                                                                --
56 -- > freq_in: El software indica los saltos que habrán en las --
57 --     tablas de señales para su generación a la frecuencia   --
58 --     especificada en el software.                               --
59 --                                                                --
60 -- > D: El software indica el ciclo de trabajo de la señal    --
61 --     a generar.                                                 --
62 --                                                                --
63 -- > volts_PC: El software indica la tensión pico a pico de la--
64 --     señal a generar.                                           --
65 --                                                                --
66 -- > offset_PC: El software indica el offset de la señal a    --
67 --     generar.                                                   --
68 --                                                                --
69 -- > position: Marca la posición del puntero al generador de  --
70 --     señales.                                                   --
71 --                                                                --
72 ------------------------------------------------------------------
```

```
73  --                      param_SG OUTPUTS                          --
74  ------------------------------------------------------------------
75  -- > freq_out: Saltos que habrán en las tablas de señales      --
76  --     para su generación a la frecuencia especificada.        --
77  --                                                              --
78  -- > ufreq: Marca las diferentes frecuencias a las que el DAC  --
79  --     muestrea.                                                --
80  --                                                              --
81  -- > volts: Indica la tensión pico a pico de la señal a        --
82  --     generar.                                                 --
83  --                                                              --
84  -- > offset: Indica el offset de la señal a generar.           --
85  --                                                              --
86  -- > freq_lcd: Frecuencia de la señal generada que se muestra --
87  --     en el display LCD cuando el control de parámetros lo    --
88  --     tiene la placa.                                          --
89  --                                                              --
90  -- > escala_lcd: Escala de la frecuencia de la señal generada --
91  --     que se muestra en el display LCD cuando el control de   --
92  --     parámetros lo tiene la placa.                            --
93  --                                                              --
94  -- > D_lcd: Ciclo de trabajo de la señal generada que se       --
95  --     muestra en el display LCD cuando el control de           --
96  --     parámetros lo tiene la placa.                            --
97  --                                                              --
98  -- > volts_lcd: Voltaje pico a pico de la señal generada que   --
99  --     se muestra en el display LCD cuando el control de       --
100 --     parámetros lo tiene la placa.                            --
101 --                                                              --
102 -- > offset_lcd: Offset de la señal generada que se muestra    --
103 --     en el display LCD cuando el control de parámetros lo    --
```

```
104 --     tiene la placa.                                          --
105 --                                                              --
106 -- > param_lcd: Selección del parámetro que se quiere          --
107 --     modificar en el display LCD cuando el control de los    --
108 --     mismos se lleva a cabo desde la placa.                   --
109 --                                                              --
110 -- > ctrl_led: Indicador led (LED<7>) del control de           --
111 --     parámetros de 'signal_generator'.                        --
112 --    - '1' -> El control del generador de señales lo tiene     --
113 --             el software.                                     --
114 --    - '0' -> El control del generador de señales lo tiene     --
115 --             la placa.                                        --
116 --                                                              --
117 ----------------------------------------------------------------
118
119 library IEEE;
120 use IEEE.STD_LOGIC_1164.ALL;
121 use IEEE.STD_LOGIC_ARITH.ALL;
122 use IEEE.STD_LOGIC_UNSIGNED.ALL;
123 use work.parametros.all;
124
125 entity param_SG is
126    port(  clk : in std_logic;
127         reset : in std_logic;
128    pc_control : in std_logic;
129  freq_mult_sw : in std_logic;
130     freq_mult : in std_logic_vector(1 downto 0);
131     f_clk_DAC : in std_logic_vector(1 downto 0);
132     sel_param : in std_logic;
133         ROT_A : in std_logic;
134         ROT_B : in std_logic;
```

```
135       freq_in : in natural range 1 to 9;
136             D : in integer range 0 to 8;
137      volts_pc : in palabra;
138     offset_pc : in palabra;
139      position : in integer range 0 to M-1;
140      freq_out : out natural range 1 to M-1;
141         ufreq : out std_logic_vector(1 downto 0);
142         volts : out palabra;
143        offset : out palabra;
144      freq_lcd : out std_logic_vector(nbits_lcd-1 downto 0);
145    escala_lcd : out std_logic_vector(1 downto 0);
146         D_lcd : out std_logic_vector(nbits_lcd-1 downto 0);
147     volts_lcd : out std_logic_vector(nbits_lcd-1 downto 0);
148    offset_lcd : out std_logic_vector(nbits_lcd downto 0);
149     param_lcd : out std_logic_vector(nbits_lcd-1 downto 0);
150      ctrl_led : out std_logic);
151 end param_SG;
152
153 architecture Behavioral of param_SG is
154
155 component rotary_switch is
156    port(  clk : in std_logic;
157         ROT_A : in std_logic;
158         ROT_B : in std_logic;
159  rotary_event : out std_logic;
160   rotary_left : out std_logic);
161 end component rotary_switch;
162
163 -- SEÑALES DEL INTERRUPTOR ROTATIVO
    ----------------------------------------------------
164 signal rotary_event, rotary_left : std_logic;
```

```
165 ----------------------------------------------------------------------
166
167 signal freq_sig : natural range 1 to 9 := 1;
168 signal escala : natural range 2 to 100 := 2;
169 signal D_sig : integer range 0 to DCycle := DCycle/2;
170 signal param : integer range 0 to 3 := 0;
171 signal ufreq_sig : std_logic_vector(1 downto 0) := (others =>
       '0');
172
173 begin
174
175 rotary_switch_unit : rotary_switch
176    PORT MAP(clk => clk,
177             ROT_A => ROT_A,
178             ROT_B => ROT_B,
179    rotary_event => rotary_event,
180     rotary_left => rotary_left);
181
182 ctrl_led <= pc_control;
183
184 -- Generación de la señal seleccionada.
185 SALTOS_TABLA : process(clk, reset)
186 begin
187    if reset = '1' then
188       freq_out <= 1;
189    elsif clk'event and clk = '1' then
190       if position > M/2 then
191          freq_out <= escala*freq_sig*D_sig/DCycle;
192       else
193          freq_out <= escala*freq_sig*(DCycle-D_sig)/DCycle;
```

```
194         end if;
195      end if;
196   end process SALTOS_TABLA;
197
198   -- Proceso para seleccionar el parámetro que queremos modificar
199   -- con el pulsador del interruptor rotativo (CENTER "R13") de
200   -- la placa cuando el control de parámetros lo tiene ésta.
201   CAMBIO_PARAMETRO : process(clk, reset, sel_param)
202      variable param_int : integer range 0 to 3 := 0;
203      variable sumar : boolean := true;
204      constant cuenta_max : natural := 5e4;
205      variable cuenta : integer range 0 to cuenta_max := 0;
206   begin
207      if reset = '1' then
208         param_int := 0;
209         sumar := true;
210         cuenta := 0;
211      elsif clk'event and clk = '1' then
212         if sel_param = '1' and sumar = true then
213            sumar := false;
214            param_int := (param_int+1);
215         elsif sel_param = '0' and sumar = false and cuenta <
               cuenta_max then
216            cuenta := cuenta+1;
217         elsif cuenta = cuenta_max then
218            sumar := true;
219            cuenta := 0;
220         else
221            param_int := param_int;
222         end if;
223      end if;
```

```
224
225 param <= param_int;
226 param_lcd <= conv_std_logic_vector(param_int,nbits_lcd);
227 end process CAMBIO_PARAMETRO;
228
229 -- Proceso que realiza el cambio de parámetros; bien si están
230 -- controlados desde la placa o desde el software.
231 EVENTOS_ROT_A_B: process(clk, reset)
232    variable D_int : integer range 0 to 8 := 4;
233    variable freq_int : natural range 1 to 13 := 1;
234    variable volts_int : natural range 1 to 9 := 1;
235    variable offset_int : integer range 0 to 30 := 15;
236 begin
237    if reset = '1' then
238       freq_sig <= 2;
239       freq_int := 1;
240       D_int := 4;
241       volts <= "000010000000";
242       volts_int := 1;
243       offset <= (others => '0');
244       offset_int := 15;
245       escala <= 2;
246    elsif clk'event and clk = '1' then
247       if pc_control = '1' then
248          freq_sig <= freq_in;
249          D_int := D;
250          offset <= offset_pc;
251          volts <= volts_pc;
252          case freq_mult is
253             when "10" =>
254                escala <= 10;
```

```
255             when "11" =>
256                escala <= 100;
257             when others =>
258                escala <= 2;
259          end case;
260       else
261          case param is
262             when 1 =>
263                if rotary_event = '1' then
264                   if rotary_left = '1' and D_int < 8 then
265                      D_int := D_int+1;
266                   elsif D_int = 0 then
267                      D_int := D_int;
268                   else
269                      D_int := D_int-1;
270                   end if;
271                end if;
272             when 2 =>
273                if rotary_event = '1' then
274                   if rotary_left = '1' and volts_int < 10 then
275                      volts_int := volts_int+1;
276                   elsif volts_int = 1 then
277                      volts_int := volts_int;
278                   else
279                      volts_int := volts_int-1;
280                   end if;
281                end if;
282             when 3 =>
283                if rotary_event = '1' then
284                   if rotary_left = '1' and offset_int < 31 then
285                      offset_int := offset_int+1;
```

```
286                     elsif offset_int = 0 then
287                        offset_int := offset_int;
288                     else
289                        offset_int := offset_int-1;
290                     end if;
291                  end if;
292               when others =>
293                  if rotary_event = '1' then
294                     if freq_mult_sw = '0' then
295                        if rotary_left = '1' and freq_int < 9 then
296                           freq_int := freq_int+1;
297                           case freq_int is
298                              when 1 => freq_sig <= 2;
299                                 escala <= 2;
300                              when 2 => freq_sig <= 3;
301                                 escala <= 2;
302                              when 3 => freq_sig <= 5;
303                                 escala <= 2;
304                              when 4 => freq_sig <= 2;
305                                 escala <= 10;
306                              when 5 => freq_sig <= 3;
307                                 escala <= 10;
308                              when 6 => freq_sig <= 4;
309                                 escala <= 10;
310                              when 7 => freq_sig <= 5;
311                                 escala <= 10;
312                              when 8 => freq_sig <= 1;
313                                 escala <= 100;
314                              when others => freq_sig <= 2;
315                                 escala <= 100;
316                           end case;
```

```
317
318                     elsif freq_int = 1 then
319                        freq_int := freq_int;
320
321                     else
322                        freq_int := freq_int-1;
323                           case freq_int is
324                              when 1 => freq_sig <= 2;
325                                 escala <= 2;
326                              when 2 => freq_sig <= 3;
327                                 escala <= 2;
328                              when 3 => freq_sig <= 5;
329                                 escala <= 2;
330                              when 4 => freq_sig <= 2;
331                                 escala <= 10;
332                              when 5 => freq_sig <= 3;
333                                 escala <= 10;
334                              when 6 => freq_sig <= 4;
335                                 escala <= 10;
336                              when 7 => freq_sig <= 5;
337                                 escala <= 10;
338                              when 8 => freq_sig <= 1;
339                                 escala <= 100;
340                              when others => freq_sig <= 2;
341                                 escala <= 100;
342                           end case;
343                     end if;
344
345                  else
346                     if rotary_left = '1' and freq_int < 12 then
347                        freq_int := freq_int+1;
```

```
348                         case freq_int is
349                            when 1 => freq_sig <= 1;
350                               escala <= 2;
351                            when 2 => freq_sig <= 2;
352                               escala <= 2;
353                            when 3 => freq_sig <= 4;
354                               escala <= 2;
355                            when 4 => freq_sig <= 8;
356                               escala <= 2;
357                            when 5 => freq_sig <= 9;
358                               escala <= 2;
359                            when 6 => freq_sig <= 3;
360                               escala <= 10;
361                            when 7 => freq_sig <= 4;
362                               escala <= 10;
363                            when 8 => freq_sig <= 8;
364                               escala <= 10;
365                            when 9 => freq_sig <= 3;
366                               escala <= 100;
367                            when 10 => freq_sig <= 2;
368                               escala <= 100;
369                            when 11 => freq_sig <= 4;
370                               escala <= 100;
371                            when others => freq_sig <= 8;
372                               escala <= 100;
373                         end case;
374
375                      elsif freq_int = 1 then
376                         freq_int := freq_int;
377
378                      else
```

```
379                         freq_int := freq_int-1;
380                         case freq_int is
381                            when 1 => freq_sig <= 1;
382                               escala <= 2;
383                            when 2 => freq_sig <= 2;
384                               escala <= 2;
385                            when 3 => freq_sig <= 4;
386                               escala <= 2;
387                            when 4 => freq_sig <= 8;
388                               escala <= 2;
389                            when 5 => freq_sig <= 9;
390                               escala <= 2;
391                            when 6 => freq_sig <= 3;
392                               escala <= 10;
393                            when 7 => freq_sig <= 4;
394                               escala <= 10;
395                            when 8 => freq_sig <= 8;
396                               escala <= 10;
397                            when 9 => freq_sig <= 3;
398                               escala <= 100;
399                            when 10 => freq_sig <= 2;
400                               escala <= 100;
401                            when 11 => freq_sig <= 4;
402                               escala <= 100;
403                            when others => freq_sig <= 8;
404                               escala <= 100;
405                         end case;
406                      end if;
407                   end if;
408                end if;
409             end case;
```

```
410
411          case volts_int is
412             when 1 => volts <= "000010000000";
413             when 2 => volts <= "000100000000";
414             when 3 => volts <= "000110000000";
415             when 4 => volts <= "001000000000";
416             when 5 => volts <= "001010000000";
417             when 6 => volts <= "001100000000";
418             when 7 => volts <= "001110000000";
419             when 8 => volts <= "010000000000";
420             when 9 => volts <= "010010000000";
421             when others => volts <= "000000000000";
422          end case;
423
424          case offset_int is
425             when 0  => offset <= "011110000000";
426             when 1  => offset <= "011100000000";
427             when 2  => offset <= "011010000000";
428             when 3  => offset <= "011000000000";
429             when 4  => offset <= "010110000000";
430             when 5  => offset <= "010100000000";
431             when 6  => offset <= "010010000000";
432             when 7  => offset <= "010000000000";
433             when 8  => offset <= "001110000000";
434             when 9  => offset <= "001100000000";
435             when 10 => offset <= "001010000000";
436             when 11 => offset <= "001000000000";
437             when 12 => offset <= "000110000000";
438             when 13 => offset <= "000100000000";
439             when 14 => offset <= "000010000000";
440             when 15 => offset <= "000000000000";
```

```
441             when 16 => offset <= "111110000000";
442             when 17 => offset <= "111100000000";
443             when 18 => offset <= "111010000000";
444             when 19 => offset <= "111000000000";
445             when 20 => offset <= "110110000000";
446             when 21 => offset <= "110100000000";
447             when 22 => offset <= "110010000000";
448             when 23 => offset <= "110000000000";
449             when 24 => offset <= "101110000000";
450             when 25 => offset <= "101100000000";
451             when 26 => offset <= "101010000000";
452             when 27 => offset <= "101000000000";
453             when 28 => offset <= "100110000000";
454             when 29 => offset <= "100100000000";
455             when 30 => offset <= "100010000000";
456             when others => offset <= "000000000000";
457          end case;
458       end if;
459
460    case D_int is
461       when 0 => D_sig <= 1024;
462       when 1 => D_sig <= 896;
463       when 2 => D_sig <= 768;
464       when 3 => D_sig <= 640;
465       when 4 => D_sig <= 512;
466       when 5 => D_sig <= 384;
467       when 6 => D_sig <= 256;
468       when 7 => D_sig <= 128;
469       when 8 => D_sig <= 0;
470       when others => D_sig <= 512;
471    end case;
```

```
472
473 end if;
474
475 -- Envío de los parámetros a la pantalla LCD.
476 freq_lcd <= conv_std_logic_vector(freq_int,nbits_lcd);
477 D_lcd <= conv_std_logic_vector(D_int,nbits_lcd);
478 volts_lcd <= conv_std_logic_vector(volts_int,nbits_lcd);
479 offset_lcd <= conv_std_logic_vector((15-offset_int),nbits_lcd+1)
    ;
480
481 end process EVENTOS_ROT_A_B;
482
483 escala_lcd <= '0'&freq_mult_sw when pc_control = '0' else
    f_clk_DAC;
484
485 ufreq_sig <= f_clk_DAC when pc_control = '1' else "00" when
    freq_mult_sw = '0' else "01";
486 ufreq <= ufreq_sig;
487
488 end Behavioral;
```

rotary_switch.vhd

```
1 ----------------------------------------------------------------------
2 -- NELSON SUÁREZ MARTÍN                        rotary_switch.vhd --
3 ----------------------------------------------------------------------
4 -- Máster en Ingeniería Industrial                                  --
5 -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual      --
6 --          para la Adquisición y Generación de Señales.          --
```

```
7  -- Fecha: Septiembre 2017.                                          --
8  -- Tutor/es: Oswaldo González & Jonas Lüke                         --
9  -- Universidad de La Laguna                                        --
10 --                                                                 --
11 ---------------------------------------------------------------------
12 --                          DESCRIPCIÓN                            --
13 ---------------------------------------------------------------------
14 -- Fichero modificado de un original de Xilinx, Inc., realiza     --
15 -- el incremento y decremento de los parámetros seleccionados     --
16 -- con el interruptor rotativo (ROT-A y ROT-B).                    --
17 --                                                                 --
18 ---------------------------------------------------------------------
19 --                   rotary_switch INPUTS                          --
20 ---------------------------------------------------------------------
21 -- > clk: Señal de reloj de la placa a 50 MHz.                     --
22 --                                                                 --
23 -- > ROT_A y ROT_B: Sentido de rotación del interruptor            --
24 --     rotativo de la placa.                                       --
25 --                                                                 --
26 ---------------------------------------------------------------------
27 --                   rotary_switch OUTPUTS                         --
28 ---------------------------------------------------------------------
29 -- > rotary_event: Indica que ha habido una rotación del           --
30 --     interruptor rotativo.                                       --
31 --                                                                 --
32 -- > rotary_left: Indica la dirección hacia la que ha rotado       --
33 --     el interruptor rotativo.                                    --
34 --                                                                 --
35 ---------------------------------------------------------------------
36
37 library IEEE;
```

```
38 use IEEE.STD_LOGIC_1164.ALL;
39
40 entity rotary_switch is
41    port(  clk : in std_logic;
42         ROT_A : in std_logic;
43         ROT_B : in std_logic;
44  rotary_event : out std_logic;
45   rotary_left : out std_logic
46 );
47 end rotary_switch;
48
49 architecture Behavioral of rotary_switch is
50
51 signal rotary_a_in, rotary_b_in, rotary_q1, rotary_q2,
      delay_rotary_q1, rotary_left_int : std_logic := '0';
52 signal rotary_in : std_logic_vector(1 downto 0) := (others =>
      '0');
53
54 begin
55
56 -- Proceso que determina si se ha realizado la rotación del
57 -- interruptor rotativo (ROT-A y ROT-B).
58 ROTARY_FILTER: process(clk)
59 begin
60    if clk'event and clk = '1' then
61       rotary_a_in <= ROT_A;
62       rotary_b_in <= ROT_B;
63       rotary_in <= rotary_b_in & rotary_a_in;
64       case rotary_in is
65          when "00" => rotary_q1 <= '0';
66                       rotary_q2 <= rotary_q2;
```

```
67
68          when "01" => rotary_q1 <= rotary_q1;
69                       rotary_q2 <= '0';
70
71          when "10" => rotary_q1 <= rotary_q1;
72                       rotary_q2 <= '1';
73
74          when "11" => rotary_q1 <= '1';
75                       rotary_q2 <= rotary_q2;
76
77          when others => rotary_q1 <= rotary_q1;
78                         rotary_q2 <= rotary_q2;
79       end case;
80    end if;
81 end process rotary_filter;
82
83 -- Proceso que determina hacia dónde se ha realizado la rotación
84 -- en el proceso anterior.
85 DIRECTION: process(clk)
86 begin
87    if clk'event and clk = '1' then
88       delay_rotary_q1 <= rotary_q1;
89       if rotary_q1 = '1' and delay_rotary_q1 = '0' then
90          rotary_event <= '1';
91          rotary_left_int <= rotary_q2;
92       else
93          rotary_event <= '0';
94          rotary_left_int <= rotary_left_int;
95       end if;
96    end if;
97 end process direction;
```

```
98
99  rotary_left <= rotary_left_int;
100
101 end Behavioral;
```

LCD.vhd

```
1   ----------------------------------------------------------------
2   -- NELSON SUÁREZ MARTÍN                                LCD.vhd --
3   ----------------------------------------------------------------
4   -- Máster en Ingeniería Industrial                             --
5   -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
6   --          para la Adquisición y Generación de Señales.      --
7   -- Fecha: Septiembre 2017.                                     --
8   -- Tutor/es: Oswaldo González & Jonas Lüke                    --
9   -- Universidad de La Laguna                                   --
10  --                                                           --
11  ----------------------------------------------------------------
12  --                                                           --
13  -- Original de "lcd.vhd -- general LCD testing program"      --
14  --     > Dan Pederson, 2004                                  --
15  --     > Barron Barnett, 2004                                --
16  --     > Jacob Beck, 2006                                    --
17  --                                                           --
18  ----------------------------------------------------------------
19  --                        DESCRIPCIÓN                        --
20  ----------------------------------------------------------------
21  -- Módulo para el control en la lectura y escritura del      --
22  -- display LCD.                                              --
23  --                                                           --
```

```
24 ----------------------------------------------------------------
25 --                        LCD COMPONENTS                      --
26 ----------------------------------------------------------------
27 -- > REFRESH_LCD: Reset cada cierto tiempo definido en el     --
28 --     paquete 'parametros.vhd' del display LCD.              --
29 --                                                            --
30 ----------------------------------------------------------------
31 --                         LCD INPUTS                         --
32 ----------------------------------------------------------------
33 -- > clk: Señal de reloj de la placa a 50 MHz.                --
34 --                                                            --
35 -- > reset: Señal de reseteo del módulo.                      --
36 --                                                            --
37 -- > sg_board: Indica si los parámetros son modificados a     --
38 --     través del software, en cuyo caso muestra un mensaje   --
39 --     en el display como que esto es así, o si estos los     --
40 --     modifica la placa, en cuyo caso se visualizan los      --
41 --     parámetros que pueden ser modificados.                 --
42 --                                                            --
43 -- > escala: Escala de la frecuencia de la señal generada     --
44 --     que se muestra en el display LCD cuando el control de  --
45 --     parámetros lo tiene la placa.                          --
46 --                                                            --
47 -- > param: Selección del parámetro que se quiere modificar   --
48 --     en el display LCD cuando el control de los mismos se   --
49 --     lleva a cabo desde la placa.                           --
50 --                                                            --
51 -- > freq: Frecuencia de la señal generada que se muestra en  --
52 --     el display LCD cuando el control de parámetros lo      --
53 --     tiene la placa.                                        --
54 --                                                            --
```

```
55 -- > D: Ciclo de trabajo de la señal generada que se muestra  --
56 --     en el display LCD cuando el control de parámetros lo   --
57 --     tiene la placa.                                        --
58 --                                                            --
59 -- > volts: Voltaje pico a pico de la señal generada que se   --
60 --     muestra en el display LCD cuando el control de         --
61 --     parámetros lo tiene la placa.                          --
62 --                                                            --
63 -- > offset: Offset de la señal generada que se muestra en el --
64 --     display LCD cuando el control de parámetros lo tiene   --
65 --     la placa.                                              --
66 --                                                            --
67 ----------------------------------------------------------------
68 --                          LCD OUTPUTS                       --
69 ----------------------------------------------------------------
70 --                                                            --
71 -- > DB: Datos de entrada/salida del display LCD de la placa. --
72 --                                                            --
73 -- > RS: Selector de registros del display LCD de la placa.   --
74 --                                                            --
75 -- > RW: Control de lectura y escritura del display LCD de    --
76 --     la placa.                                              --
77 --                                                            --
78 -- > E: Habilitar/Deshabilitar la lectura y escritura del     --
79 --     display LCD de la placa.                               --
80 --                                                            --
81 ----------------------------------------------------------------
82
83 library IEEE;
84 use IEEE.STD_LOGIC_1164.ALL;
85 use IEEE.STD_LOGIC_ARITH.ALL;
```

```
86  use IEEE.STD_LOGIC_UNSIGNED.ALL;
87  use work.parametros.all;
88
89  entity LCD is
90     port( clk : in std_logic;
91           reset : in std_logic;
92       sg_board : in std_logic;
93         escala : in std_logic_vector(1 downto 0);
94          param : in std_logic_vector(nbits_lcd-1 downto 0);
95           freq : in std_logic_vector(nbits_lcd-1 downto 0);
96              D : in std_logic_vector(nbits_lcd-1 downto 0);
97          volts : in std_logic_vector(nbits_lcd-1 downto 0);
98         offset : in std_logic_vector(nbits_lcd downto 0);
99             DB : out std_logic_vector(nbits_com-1 downto 0);
100            RS : out std_logic;
101            RW : out std_logic;
102             E : out std_logic);
103 end LCD;
104
105 architecture Behavioral of LCD is
106
107 component refresh_lcd is
108    port( clk : in  std_logic;
109          reset : in  std_logic;
110            rst : out  std_logic);
111 end component refresh_lcd;
112
113 -- SEÑALES DE REFRESH_LCD ---------------------------------------
114 signal rst : std_logic := '0';
115 ------------------------------------------------------------------
116
```

```
117 --  LCD control state machine
118 type mstate is (
119 stFunctionSet,           -- Initialization states
120 stDisplayCtrlSet,
121 stDisplayClear,
122 stPowerOn_Delay,         -- Delay states
123 stFunctionSet_Delay,
124 stDisplayCtrlSet_Delay,
125 stDisplayClear_Delay,
126 stInitDne,               -- Display charachters and perform
       standard operations
127 stActWr,
128 stCharDelay              -- Write delay for operations
129 --stWait                 -- Idle state
130 );
131
132 -- Write control state machine
133 type wstate is (
134 stRW,     -- set up RS and RW
135 stEnable, -- set up E
136 stIdle    -- Write data on DB(0)-DB(7)
137 );
138
139 ----------------------------------------------------------------
140 --  Signal Declarations and Constants.                        --
141 ----------------------------------------------------------------
142 -- These constants are used to initialize the LCD pannel.
143 -- FunctionSet:
144 -- > Bit 0 and 1 are arbitrary
145 -- > Bit 2:  Displays font type(0=5x8, 1=5x11)
146 -- > Bit 3:  Numbers of display lines (0=1, 1=2)
```

```
147 -- > Bit 4:  Data length (0=4 bit, 1=8 bit)
148 -- > Bit 5-7 are set
149 -- DisplayCtrlSet:
150 -- > Bit 0:  Blinking cursor control (0=off, 1=on)
151 -- > Bit 1:  Cursor (0=off, 1=on)
152 -- > Bit 2:  Display (0=off, 1=on)
153 -- > Bit 3-7 are set
154 -- DisplayClear:
155 -- > Bit 1-7 are set
156 signal clkCount:std_logic_vector(5 downto 0);
157 signal activateW:std_logic:= '0'; -- Activate Write sequence
158 --15 bit count variable for timing delays
159 signal count:std_logic_vector (16 downto 0):=
       "00000000000000000";
160 signal delayOK:std_logic:= '0';   -- High when count has reached
        the right delay time
161 signal OneUSclk:std_logic;        -- Signal is treated as a 1
       MHz clock
162 signal stCur:mstate:= stPowerOn_Delay; -- LCD control state
       machine
163 signal stNext:mstate;
164 signal stCurW:wstate:= stIdle; -- Write control state machine
165 signal stNextW:wstate;
166 signal writeDone:std_logic:= '0'; -- Command set finish
167
168 ------------------------------------------------------------------
169 -- Texto por defecto en la pantalla LCD.                        --
170 ------------------------------------------------------------------
171 type LCD_CMDS_T is array(integer range 0 to 59) of
       std_logic_vector(9 downto 0);
172 signal LCD_CMDS : LCD_CMDS_T := (
```

```
173         0 => "00"&X"3C",  -- Function Set (b00111100)
174         1 => "00"&X"0C",  -- Display ON, Cursor OFF, Blink OFF (
                b00001100)
175         2 => "00"&X"01",  -- Clear Display
176         3 => "00"&X"02",  -- return home
177   ---------------------- COMIENZO 1ª LÍNEA ----------------------
178         4 => "10"&X"20",  -- space
179         5 => "10"&X"66",  -- f
180         6 => "10"&X"3D",  -- =
181         7 => "10"&X"20",  -- space
182         8 => "00"&X"20",
183         9 => "10"&X"20",
184        10 => "10"&X"20",
185        11 => "10"&X"20", -- space
186        12 => "10"&X"48", -- H
187        13 => "10"&X"7A", -- z
188        14 => "10"&X"20", -- space
189        15 => "10"&X"44", -- D
190        16 => "10"&X"3D", -- =
191        17 => "10"&X"20", -- space
192        18 => "10"&X"2F", -- /
193        19 => "10"&X"20", -- space
194   ------------------------ FINAL 1ª LÍNEA ------------------------
195        20 => "10"&X"18", -- cancel
196        21 => "10"&X"18",
197        22 => "10"&X"18",
198        23 => "10"&X"18",
199        24 => "10"&X"18",
200        25 => "10"&X"18",
201        26 => "10"&X"18",
202        27 => "10"&X"18",
```

```
203       28 => "10"&X"18",
204       29 => "10"&X"18",
205       30 => "10"&X"18",
206       31 => "10"&X"18",
207       32 => "10"&X"18",
208       33 => "10"&X"18",
209       34 => "10"&X"18",
210       35 => "10"&X"18",
211       36 => "10"&X"18",
212       37 => "10"&X"18",
213       38 => "10"&X"18",
214       39 => "10"&X"18",
215       40 => "10"&X"18",
216       41 => "10"&X"18",
217       42 => "10"&X"18",
218       43 => "10"&X"18", -- cancel
219  ---------------------- COMIENZO 2ª LÍNEA -----------------------
220       44 => "10"&X"20", -- space
221       45 => "10"&X"56", -- V
222       46 => "10"&X"3D", -- =
223       47 => "10"&X"20", -- space
224       48 => "10"&X"2E", -- .
225       49 => "10"&X"20", -- space
226       50 => "10"&X"20",
227       51 => "10"&X"20",
228       52 => "10"&X"30", -- 0
229       53 => "10"&X"2E", -- .
230       54 => "10"&X"20", -- space
231       55 => "10"&X"20", -- space
232       56 => "10"&X"56", -- V
233       57 => "10"&X"20", -- space
```

```
234       58 => "10"&X"20", -- space
235       59 => "01"&X"18");-- cancel
236   ----------------------------------------------------------------
237
238   signal lcd_cmd_ptr : integer range 0 to LCD_CMDS'HIGH + 1 := 0;
239
240   begin
241
242   refresh_lcd_unit : refresh_lcd
243      PORT MAP(clk => clk,
244               reset => reset,
245                 rst => rst);
246
247   ----------------------------------------------------------------
248   -- Proceso para escribir los datos en la pantalla LCD.
249   ----------------------------------------------------------------
250   write_lcd :     process (clk, sg_board, rst)
251   begin
252      if rst = '1' then
253         LCD_CMDS(4) <= "10"&X"20";
254         LCD_CMDS(5) <= "10"&X"20";
255         LCD_CMDS(6) <= "10"&X"20";
256         LCD_CMDS(7) <= "10"&X"20";
257         LCD_CMDS(8) <= "10"&X"20";
258         LCD_CMDS(9) <= "10"&X"20";
259         LCD_CMDS(10) <= "10"&X"20";
260         LCD_CMDS(11) <= "10"&X"20";
261         LCD_CMDS(12) <= "10"&X"20";
262         LCD_CMDS(13) <= "10"&X"20";
263         LCD_CMDS(14) <= "10"&X"20";
264         LCD_CMDS(15) <= "10"&X"20";
```

```
265         LCD_CMDS(16) <= "10"&X"20";
266         LCD_CMDS(17) <= "10"&X"20";
267         LCD_CMDS(18) <= "10"&X"20";
268         LCD_CMDS(19) <= "10"&X"20";
269         LCD_CMDS(20) <= "10"&X"18";
270         LCD_CMDS(21) <= "10"&X"18";
271         LCD_CMDS(22) <= "10"&X"18";
272         LCD_CMDS(23) <= "10"&X"18";
273         LCD_CMDS(24) <= "10"&X"18";
274         LCD_CMDS(25) <= "10"&X"18";
275         LCD_CMDS(26) <= "10"&X"18";
276         LCD_CMDS(27) <= "10"&X"18";
277         LCD_CMDS(28) <= "10"&X"18";
278         LCD_CMDS(29) <= "10"&X"18";
279         LCD_CMDS(30) <= "10"&X"18";
280         LCD_CMDS(31) <= "10"&X"18";
281         LCD_CMDS(32) <= "10"&X"18";
282         LCD_CMDS(33) <= "10"&X"18";
283         LCD_CMDS(34) <= "10"&X"18";
284         LCD_CMDS(35) <= "10"&X"18";
285         LCD_CMDS(36) <= "10"&X"18";
286         LCD_CMDS(37) <= "10"&X"18";
287         LCD_CMDS(38) <= "10"&X"18";
288         LCD_CMDS(39) <= "10"&X"18";
289         LCD_CMDS(40) <= "10"&X"18";
290         LCD_CMDS(41) <= "10"&X"18";
291         LCD_CMDS(42) <= "10"&X"18";
292         LCD_CMDS(43) <= "10"&X"18";
293         LCD_CMDS(44) <= "10"&X"20";
294         LCD_CMDS(45) <= "10"&X"20";
295         LCD_CMDS(46) <= "10"&X"20";
```

```
296         LCD_CMDS(47) <= "10"&X"20";
297         LCD_CMDS(48) <= "10"&X"20";
298         LCD_CMDS(49) <= "10"&X"20";
299         LCD_CMDS(50) <= "10"&X"20";
300         LCD_CMDS(51) <= "10"&X"20";
301         LCD_CMDS(52) <= "10"&X"20";
302         LCD_CMDS(53) <= "10"&X"20";
303         LCD_CMDS(54) <= "10"&X"20";
304         LCD_CMDS(55) <= "10"&X"20";
305         LCD_CMDS(56) <= "10"&X"20";
306         LCD_CMDS(57) <= "10"&X"20";
307         LCD_CMDS(58) <= "10"&X"20";
308         LCD_CMDS(59) <= "10"&X"18";
309
310      elsif sg_board = '1' then
311         LCD_CMDS(4) <= "10"&X"43";  -- C
312         LCD_CMDS(5) <= "10"&X"6F";  -- o
313         LCD_CMDS(6) <= "10"&X"6E";  -- n
314         LCD_CMDS(7) <= "10"&X"74";  -- t
315         LCD_CMDS(8) <= "10"&X"72";  -- r
316         LCD_CMDS(9) <= "10"&X"6F";  -- o
317         LCD_CMDS(10) <= "10"&X"6C"; -- l
318         LCD_CMDS(11) <= "10"&X"20"; -- space
319         LCD_CMDS(12) <= "10"&X"64"; -- d
320         LCD_CMDS(13) <= "10"&X"65"; -- e
321         LCD_CMDS(14) <= "10"&X"20"; -- space
322         LCD_CMDS(15) <= "10"&X"53"; -- S
323         LCD_CMDS(16) <= "10"&X"2E"; -- .
324         LCD_CMDS(17) <= "10"&X"47"; -- G
325         LCD_CMDS(18) <= "10"&X"2E"; -- .
326         LCD_CMDS(19) <= "10"&X"20"; -- space
```

```
327
328         LCD_CMDS(44) <= "10"&X"70"; -- p
329         LCD_CMDS(45) <= "10"&X"6F"; -- o
330         LCD_CMDS(46) <= "10"&X"72"; -- r
331         LCD_CMDS(47) <= "10"&X"20"; -- space
332         LCD_CMDS(48) <= "10"&X"73"; -- s
333         LCD_CMDS(49) <= "10"&X"6F"; -- o
334         LCD_CMDS(50) <= "10"&X"66"; -- f
335         LCD_CMDS(51) <= "10"&X"74"; -- t
336         LCD_CMDS(52) <= "10"&X"77"; -- w
337         LCD_CMDS(53) <= "10"&X"61"; -- a
338         LCD_CMDS(54) <= "10"&X"72"; -- r
339         LCD_CMDS(55) <= "10"&X"65"; -- e
340         LCD_CMDS(56) <= "10"&X"21"; -- !
341         LCD_CMDS(57) <= "10"&X"21"; -- !
342         LCD_CMDS(58) <= "10"&X"20"; -- space
343      elsif clk'event and clk = '1' then
344         if param = "0000" then
345            LCD_CMDS(4)  <= "10"&X"3E"; -- >
346         else
347            LCD_CMDS(4)  <= "10"&X"20"; -- space
348         end if;
349
350         LCD_CMDS(5)  <= "10"&X"66"; -- f
351         LCD_CMDS(6)  <= "10"&X"3D"; -- =
352
353         case freq is
354            when "0001" =>
355               LCD_CMDS(7) <= "10"&X"30"; -- 0
356               LCD_CMDS(8) <= "10"&X"2E"; -- .
357               if escala = "00" then
```

```
358                LCD_CMDS(9) <= "10"&X"31";  -- 1
359                LCD_CMDS(10) <= "10"&X"35"; -- 5
360             else
361                LCD_CMDS(9) <= "10"&X"30";  -- 0
362                LCD_CMDS(10) <= "10"&X"31"; -- 1
363             end if;
364          when "0010" =>
365             LCD_CMDS(7) <= "10"&X"30"; -- 0
366             LCD_CMDS(8) <= "10"&X"2E"; -- .
367             if escala = "00" then
368                LCD_CMDS(9) <= "10"&X"32";  -- 2
369                LCD_CMDS(10) <= "10"&X"35"; -- 5
370             else
371                LCD_CMDS(9) <= "10"&X"30";  -- 0
372                LCD_CMDS(10) <= "10"&X"32"; -- 2
373             end if;
374          when "0011" =>
375             LCD_CMDS(7) <= "10"&X"30"; -- 0
376             LCD_CMDS(8) <= "10"&X"2E"; -- .
377             if escala = "00" then
378                LCD_CMDS(9) <= "10"&X"35";  -- 5
379                LCD_CMDS(10) <= "10"&X"30"; -- 0
380             else
381                LCD_CMDS(9) <= "10"&X"30";  -- 0
382                LCD_CMDS(10) <= "10"&X"35"; -- 5
383             end if;
384          when "0100" =>
385             if escala = "00" then
386                LCD_CMDS(7) <= "10"&X"31";  -- 1
387                LCD_CMDS(8) <= "10"&X"2E";  -- .
388                LCD_CMDS(9) <= "10"&X"30";  -- 0
```

```
389             else
390                LCD_CMDS(7) <= "10"&X"30";  -- 0
391                LCD_CMDS(8) <= "10"&X"2E";  -- .
392                LCD_CMDS(9) <= "10"&X"31";  -- 1
393             end if;
394                LCD_CMDS(10) <= "10"&X"30"; -- 0
395          when "0101" =>
396             if escala = "00" then
397                LCD_CMDS(7) <= "10"&X"31";  -- 1
398                LCD_CMDS(8) <= "10"&X"2E";  -- .
399                LCD_CMDS(9) <= "10"&X"32";  -- 2
400                LCD_CMDS(10) <= "10"&X"35"; -- 5
401             else
402                LCD_CMDS(7) <= "10"&X"30";  -- 0
403                LCD_CMDS(8) <= "10"&X"2E";  -- .
404                LCD_CMDS(9) <= "10"&X"31";  -- 1
405                LCD_CMDS(10) <= "10"&X"32"; -- 2
406             end if;
407          when "0110" =>
408             if escala = "00" then
409                LCD_CMDS(7) <= "10"&X"32"; -- 2
410                LCD_CMDS(8) <= "10"&X"2E"; -- .
411                LCD_CMDS(9) <= "10"&X"30"; -- 0
412             else
413                LCD_CMDS(7) <= "10"&X"30"; -- 0
414                LCD_CMDS(8) <= "10"&X"2E"; -- .
415                LCD_CMDS(9) <= "10"&X"32"; -- 2
416             end if;
417                LCD_CMDS(10) <= "10"&X"30"; -- 0
418          when "0111" =>
419             if escala = "00" then
```

```
420                  LCD_CMDS(7) <= "10"&X"32";  -- 2
421                  LCD_CMDS(8) <= "10"&X"2E";  -- .
422                  LCD_CMDS(9) <= "10"&X"35";  -- 5
423                  LCD_CMDS(10) <= "10"&X"30"; -- 0
424               else
425                  LCD_CMDS(7) <= "10"&X"30";  -- 0
426                  LCD_CMDS(8) <= "10"&X"2E";  -- .
427                  LCD_CMDS(9) <= "10"&X"32";  -- 2
428                  LCD_CMDS(10) <= "10"&X"35"; -- 5
429               end if;
430            when "1000" =>
431               if escala = "00" then
432                  LCD_CMDS(7) <= "10"&X"35"; -- 5
433                  LCD_CMDS(8) <= "10"&X"2E"; -- .
434                  LCD_CMDS(9) <= "10"&X"30"; -- 0
435               else
436                  LCD_CMDS(7) <= "10"&X"30"; -- 0
437                  LCD_CMDS(8) <= "10"&X"2E"; -- .
438                  LCD_CMDS(9) <= "10"&X"35"; -- 5
439               end if;
440                  LCD_CMDS(10) <= "10"&X"30";   -- 0
441            when "1001" =>
442               LCD_CMDS(7) <= "10"&X"31";    -- 1
443               if escala = "00" then
444                  LCD_CMDS(8) <= "10"&X"30"; -- 0
445                  LCD_CMDS(9) <= "10"&X"2E"; -- .
446               else
447                  LCD_CMDS(8) <= "10"&X"2E"; -- .
448                  LCD_CMDS(9) <= "10"&X"30"; -- 0
449               end if;
450               LCD_CMDS(10) <= "10"&X"30";   -- 0
```

```
451            when "1010" =>
452               LCD_CMDS(7)  <= "10"&X"31"; -- 1
453               LCD_CMDS(8)  <= "10"&X"2E"; -- .
454               LCD_CMDS(9)  <= "10"&X"32"; -- 2
455               LCD_CMDS(10) <= "10"&X"35"; -- 5
456            when "1011" =>
457               LCD_CMDS(7)  <= "10"&X"32"; -- 2
458               LCD_CMDS(8)  <= "10"&X"2E"; -- .
459               LCD_CMDS(9)  <= "10"&X"35"; -- 5
460               LCD_CMDS(10) <= "10"&X"30"; -- 0
461            when "1100" =>
462               LCD_CMDS(7)  <= "10"&X"35"; -- 5
463               LCD_CMDS(8)  <= "10"&X"2E"; -- .
464               LCD_CMDS(9)  <= "10"&X"30"; -- 0
465               LCD_CMDS(10) <= "10"&X"30"; -- 0
466            when others =>
467               LCD_CMDS(7)  <= "10"&X"3F"; -- ?
468               LCD_CMDS(8)  <= "10"&X"3F"; --
469               LCD_CMDS(9)  <= "10"&X"3F"; --
470               LCD_CMDS(10) <= "10"&X"3F"; -- ?
471         end case;
472
473         if escala = "00" then
474            LCD_CMDS(11) <= "10"&X"4B"; -- K
475         else
476            LCD_CMDS(11) <= "10"&X"4D"; -- M
477         end if;
478
479         LCD_CMDS(12) <= "10"&X"48"; -- H
480         LCD_CMDS(13) <= "10"&X"7A"; -- z
481
```

```
482         if param = "0001" then
483            LCD_CMDS(14) <= "10"&X"3E"; -- >
484         else
485            LCD_CMDS(14) <= "10"&X"20"; -- space
486         end if;
487
488         LCD_CMDS(15) <= "10"&X"44"; -- D
489         LCD_CMDS(16) <= "10"&X"3D"; -- =
490
491         case D is
492            when "0000" =>
493               LCD_CMDS(17) <= "10"&X"20"; -- space
494               LCD_CMDS(18) <= "10"&X"30"; -- 0
495               LCD_CMDS(19) <= "10"&X"20"; -- space
496            when "0001" =>
497               LCD_CMDS(17) <= "10"&X"31"; -- 1
498               LCD_CMDS(18) <= "10"&X"2F"; -- /
499               LCD_CMDS(19) <= "10"&X"38"; -- 8
500            when "0010" =>
501               LCD_CMDS(17) <= "10"&X"31"; -- 1
502               LCD_CMDS(18) <= "10"&X"2F"; -- /
503               LCD_CMDS(19) <= "10"&X"34"; -- 4
504            when "0011" =>
505               LCD_CMDS(17) <= "10"&X"33"; -- 3
506               LCD_CMDS(18) <= "10"&X"2F"; -- /
507               LCD_CMDS(19) <= "10"&X"38"; -- 8
508            when "0100" =>
509               LCD_CMDS(17) <= "10"&X"31"; -- 1
510               LCD_CMDS(18) <= "10"&X"2F"; -- /
511               LCD_CMDS(19) <= "10"&X"32"; -- 2
512            when "0101" =>
```

```
513             LCD_CMDS(17) <= "10"&X"35"; -- 5
514             LCD_CMDS(18) <= "10"&X"2F"; -- /
515             LCD_CMDS(19) <= "10"&X"38"; -- 8
516          when "0110" =>
517             LCD_CMDS(17) <= "10"&X"33"; -- 3
518             LCD_CMDS(18) <= "10"&X"2F"; -- /
519             LCD_CMDS(19) <= "10"&X"34"; -- 4
520          when "0111" =>
521             LCD_CMDS(17) <= "10"&X"37"; -- 7
522             LCD_CMDS(18) <= "10"&X"2F"; -- /
523             LCD_CMDS(19) <= "10"&X"38"; -- 8
524          when "1000" =>
525             LCD_CMDS(17) <= "10"&X"20"; -- space
526             LCD_CMDS(18) <= "10"&X"31"; -- 1
527             LCD_CMDS(19) <= "10"&X"20"; -- space
528          when others =>
529             LCD_CMDS(17) <= "10"&X"20"; -- space
530             LCD_CMDS(18) <= "10"&X"3F"; -- ?
531             LCD_CMDS(19) <= "10"&X"20"; -- space
532       end case;
533
534 ---------------------- COMIENZO 2ª LÍNEA -----------------------
535
536       if param = "0010" then
537          LCD_CMDS(44) <= "10"&X"3E"; -- >
538       else
539          LCD_CMDS(44) <= "10"&X"20"; -- space
540       end if;
541
542       LCD_CMDS(45) <= "10"&X"56"; -- V
543       LCD_CMDS(46) <= "10"&X"3D"; -- =
```

```
544
545         case volts is
546            when "0001" =>
547               LCD_CMDS(47) <= "10"&X"30"; -- 0
548               LCD_CMDS(48) <= "10"&X"2E"; -- .
549               LCD_CMDS(49) <= "10"&X"32"; -- 2
550               LCD_CMDS(50) <= "10"&X"35"; -- 5
551            when "0010" =>
552               LCD_CMDS(47) <= "10"&X"30"; -- 0
553               LCD_CMDS(48) <= "10"&X"2E"; -- .
554               LCD_CMDS(49) <= "10"&X"35"; -- 5
555               LCD_CMDS(50) <= "10"&X"30"; -- 0
556            when "0011" =>
557               LCD_CMDS(47) <= "10"&X"30"; -- 0
558               LCD_CMDS(48) <= "10"&X"2E"; -- .
559               LCD_CMDS(49) <= "10"&X"37"; -- 7
560               LCD_CMDS(50) <= "10"&X"35"; -- 0
561            when "0100" =>
562               LCD_CMDS(47) <= "10"&X"30"; -- 0
563               LCD_CMDS(48) <= "10"&X"2E"; -- .
564               LCD_CMDS(49) <= "10"&X"39"; -- 9
565               LCD_CMDS(50) <= "10"&X"30"; -- 0
566            when "0101" =>
567               LCD_CMDS(47) <= "10"&X"31"; -- 1
568               LCD_CMDS(48) <= "10"&X"2E"; -- .
569               LCD_CMDS(49) <= "10"&X"30"; -- 0
570               LCD_CMDS(50) <= "10"&X"30"; -- 0
571            when "0110" =>
572               LCD_CMDS(47) <= "10"&X"31"; -- 1
573               LCD_CMDS(48) <= "10"&X"2E"; -- .
574               LCD_CMDS(49) <= "10"&X"32"; -- 2
```

```
575             LCD_CMDS(50) <= "10"&X"35"; -- 5
576          when "0111" =>
577             LCD_CMDS(47) <= "10"&X"31"; -- 1
578             LCD_CMDS(48) <= "10"&X"2E"; -- .
579             LCD_CMDS(49) <= "10"&X"35"; -- 5
580             LCD_CMDS(50) <= "10"&X"30"; -- 0
581          when "1000" =>
582             LCD_CMDS(47) <= "10"&X"31"; -- 1
583             LCD_CMDS(48) <= "10"&X"2E"; -- .
584             LCD_CMDS(49) <= "10"&X"37"; -- 7
585             LCD_CMDS(50) <= "10"&X"30"; -- 0
586          when "1001" =>
587             LCD_CMDS(47) <= "10"&X"32"; -- 2
588             LCD_CMDS(48) <= "10"&X"2E"; -- .
589             LCD_CMDS(49) <= "10"&X"30"; -- 0
590             LCD_CMDS(50) <= "10"&X"30"; -- 0
591          when others =>
592             LCD_CMDS(47) <= "10"&X"3F"; -- ?
593             LCD_CMDS(48) <= "10"&X"3F";
594             LCD_CMDS(49) <= "10"&X"3F";
595             LCD_CMDS(50) <= "10"&X"3F"; -- ?
596       end case;
597
598       if offset > "10000" or offset = "00000" then
599          LCD_CMDS(51) <= "10"&X"2B"; -- +
600       else
601          LCD_CMDS(51) <= "10"&X"2D"; -- -
602       end if;
603
604       LCD_CMDS(52) <= "10"&X"30"; -- 0
605       LCD_CMDS(53) <= "10"&X"2E"; -- .
```

```
606
607         case offset is
608            when "00000" =>
609               LCD_CMDS(54)  <= "10"&X"30"; -- 0
610               LCD_CMDS(55)  <= "10"&X"30"; -- 0
611            when "00001" =>
612               LCD_CMDS(54)  <= "10"&X"30"; -- 0
613               LCD_CMDS(55)  <= "10"&X"35"; -- 5
614            when "00010" =>
615               LCD_CMDS(54)  <= "10"&X"31"; -- 1
616               LCD_CMDS(55)  <= "10"&X"30"; -- 0
617            when "00011" =>
618               LCD_CMDS(54)  <= "10"&X"31"; -- 1
619               LCD_CMDS(55)  <= "10"&X"35"; -- 5
620            when "00100" =>
621               LCD_CMDS(54)  <= "10"&X"32"; -- 2
622               LCD_CMDS(55)  <= "10"&X"30"; -- 0
623            when "00101" =>
624               LCD_CMDS(54)  <= "10"&X"32"; -- 2
625               LCD_CMDS(55)  <= "10"&X"35"; -- 5
626            when "00110" =>
627               LCD_CMDS(54)  <= "10"&X"33"; -- 3
628               LCD_CMDS(55)  <= "10"&X"30"; -- 0
629            when "00111" =>
630               LCD_CMDS(54)  <= "10"&X"33"; -- 3
631               LCD_CMDS(55)  <= "10"&X"35"; -- 5
632            when "01000" =>
633               LCD_CMDS(54)  <= "10"&X"34"; -- 4
634               LCD_CMDS(55)  <= "10"&X"30"; -- 0
635            when "01001" =>
636               LCD_CMDS(54)  <= "10"&X"34"; -- 4
```

```
637             LCD_CMDS(55)  <= "10"&X"35"; -- 5
638          when "01010" =>
639             LCD_CMDS(54)  <= "10"&X"35"; -- 5
640             LCD_CMDS(55)  <= "10"&X"30"; -- 0
641          when "01011" =>
642             LCD_CMDS(54)  <= "10"&X"35"; -- 5
643             LCD_CMDS(55)  <= "10"&X"35"; -- 5
644          when "01100" =>
645             LCD_CMDS(54)  <= "10"&X"36"; -- 6
646             LCD_CMDS(55)  <= "10"&X"30"; -- 0
647          when "01101" =>
648             LCD_CMDS(54)  <= "10"&X"36"; -- 6
649             LCD_CMDS(55)  <= "10"&X"35"; -- 5
650          when "01110" =>
651             LCD_CMDS(54)  <= "10"&X"37"; -- 7
652             LCD_CMDS(55)  <= "10"&X"30"; -- 0
653          when "01111" =>
654             LCD_CMDS(54)  <= "10"&X"37"; -- 7
655             LCD_CMDS(55)  <= "10"&X"35"; -- 5
656          when "10001" =>
657             LCD_CMDS(54)  <= "10"&X"37"; -- 7
658             LCD_CMDS(55)  <= "10"&X"35"; -- 5
659          when "10010" =>
660             LCD_CMDS(54)  <= "10"&X"37"; -- 7
661             LCD_CMDS(55)  <= "10"&X"30"; -- 0
662          when "10011" =>
663             LCD_CMDS(54)  <= "10"&X"36"; -- 6
664             LCD_CMDS(55)  <= "10"&X"35"; -- 5
665          when "10100" =>
666             LCD_CMDS(54)  <= "10"&X"36"; -- 6
667             LCD_CMDS(55)  <= "10"&X"30"; -- 0
```

```
668             when "10101" =>
669                LCD_CMDS(54)  <= "10"&X"35"; -- 5
670                LCD_CMDS(55)  <= "10"&X"35"; -- 5
671             when "10110" =>
672                LCD_CMDS(54)  <= "10"&X"35"; -- 5
673                LCD_CMDS(55)  <= "10"&X"30"; -- 0
674             when "10111" =>
675                LCD_CMDS(54)  <= "10"&X"34"; -- 4
676                LCD_CMDS(55)  <= "10"&X"35"; -- 5
677             when "11000" =>
678                LCD_CMDS(54)  <= "10"&X"34"; -- 4
679                LCD_CMDS(55)  <= "10"&X"30"; -- 0
680             when "11001" =>
681                LCD_CMDS(54)  <= "10"&X"33"; -- 3
682                LCD_CMDS(55)  <= "10"&X"35"; -- 5
683             when "11010" =>
684                LCD_CMDS(54)  <= "10"&X"33"; -- 3
685                LCD_CMDS(55)  <= "10"&X"30"; -- 0
686             when "11011" =>
687                LCD_CMDS(54)  <= "10"&X"32"; -- 2
688                LCD_CMDS(55)  <= "10"&X"35"; -- 5
689             when "11100" =>
690                LCD_CMDS(54)  <= "10"&X"32"; -- 2
691                LCD_CMDS(55)  <= "10"&X"30"; -- 0
692             when "11101" =>
693                LCD_CMDS(54)  <= "10"&X"31"; -- 1
694                LCD_CMDS(55)  <= "10"&X"35"; -- 5
695             when "11110" =>
696                LCD_CMDS(54)  <= "10"&X"31"; -- 1
697                LCD_CMDS(55)  <= "10"&X"30"; -- 0
698             when "11111" =>
```

```
699             LCD_CMDS(54)  <= "10"&X"30"; -- 0
700             LCD_CMDS(55)  <= "10"&X"35"; -- 5
701          when others =>
702             LCD_CMDS(54)  <= "10"&X"3F"; -- ?
703             LCD_CMDS(55)  <= "10"&X"3F"; -- ?
704       end case;
705
706       LCD_CMDS(56)  <= "10"&X"56"; -- V
707
708       if param = "0011" then
709          LCD_CMDS(57) <= "10"&X"3C"; -- <
710       else
711          LCD_CMDS(57) <= "10"&X"20"; -- space
712       end if;
713
714    end if;
715 end process;
716 ----------------------------------------------------------------
717 ----------------------------------------------------------------
718 -- This process counts to 50, and then resets. It is used to
719 -- divide the clock signal time.
720 process (clk, oneUSclk)
721 begin
722    if (clk = '1' and clk'event) then
723       clkCount <= clkCount + 1;
724    end if;
725 end process;
726 -- This makes oneUSClock peak once every 1 microsecond
727 oneUSclk <= clkCount(0);
728
```

```
729 -- This process incriments the count variable unless delayOK =
      1.
730 process (oneUSclk, delayOK)
731 begin
732    if (oneUSclk = '1' and oneUSclk'event) then
733       if delayOK = '1' then
734 -- Velocidad de refresco de la pantalla LCD.
735          count <= "0111111111111111";
736       else
737          count <= count + 1;
738       end if;
739    end if;
740 end process;
741
742 -- This goes high when all commands have been run
743 writeDone <= '1' when (lcd_cmd_ptr = LCD_CMDS'HIGH) else '0';
744 --rdone <= '1' when stCur = stWait else '0';
745
746 -- Increments the pointer so the statemachine goes through the
747 -- commands
748 process (lcd_cmd_ptr, oneUSclk)
749 begin
750    if (oneUSclk = '1' and oneUSclk'event) then
751       if ((stNext = stInitDne or stNext = stDisplayCtrlSet or
          stNext = stDisplayClear) and writeDone = '0') then
752          lcd_cmd_ptr <= lcd_cmd_ptr + 1;
753       elsif stCur = stPowerOn_Delay or stNext = stPowerOn_Delay
          then
754          lcd_cmd_ptr <= 0;
755       else
756          lcd_cmd_ptr <= lcd_cmd_ptr;
```

```
757         end if;
758      end if;
759   end process;
760
761   -- Determines when count has gotten to the right number,
         depending on the state.
762
763   delayOK <= '1' when ((stCur = stPowerOn_Delay and count =
         "00100111001010010") or
764   (stCur = stFunctionSet_Delay and count = "00000000000110010") or
765   (stCur = stDisplayCtrlSet_Delay and count = "00000000000110010")
          or
766   (stCur = stDisplayClear_Delay and count = "00000011001000000")
         or
767   -- Max Delay for character writes and shifts
768   (stCur = stCharDelay and count = "1111111111111111"))
769   -- 37  This is proper delay between writes to ram.
770   else    '0';
771
772   -- This process runs the LCD status state machine
773   process (oneUSclk, rst)
774   begin
775      if oneUSclk = '1' and oneUSclk'Event then
776         if rst = '1' then
777            stCur <= stPowerOn_Delay;
778         else
779            stCur <= stNext;
780         end if;
781      end if;
782   end process;
783
```

```
784 -- This process generates the sequence of outputs needed to
785 -- initialize and write to the LCD screen.
786 process (stCur, delayOK, writeDone, lcd_cmd_ptr)
787 begin
788    case stCur is
789 -- Delays the state machine for 20ms which is needed for
790 -- proper startup.
791       when stPowerOn_Delay =>
792          if delayOK = '1' then
793             stNext <= stFunctionSet;
794          else
795             stNext <= stPowerOn_Delay;
796          end if;
797          RS <= LCD_CMDS(lcd_cmd_ptr)(9);
798          RW <= LCD_CMDS(lcd_cmd_ptr)(8);
799          DB <= LCD_CMDS(lcd_cmd_ptr)(7 downto 0);
800          activateW <= '0';
801
802 -- This issuse the function set to the LCD as follows
803 -- 8 bit data length, 2 lines, font is 5x8.
804       when stFunctionSet =>
805          RS <= LCD_CMDS(lcd_cmd_ptr)(9);
806          RW <= LCD_CMDS(lcd_cmd_ptr)(8);
807          DB <= LCD_CMDS(lcd_cmd_ptr)(7 downto 0);
808          activateW <= '1';
809          stNext <= stFunctionSet_Delay;
810
811 -- Gives the proper delay of 37us between the function set and
812 -- the display control set.
813       when stFunctionSet_Delay =>
814          RS <= LCD_CMDS(lcd_cmd_ptr)(9);
```

```
815           RW <= LCD_CMDS(lcd_cmd_ptr)(8);
816           DB <= LCD_CMDS(lcd_cmd_ptr)(7 downto 0);
817           activateW <= '0';
818           if delayOK = '1' then
819              stNext <= stDisplayCtrlSet;
820           else
821              stNext <= stFunctionSet_Delay;
822           end if;
823
824  -- Issuse the display control set as follows
825  -- Display ON,  Cursor OFF, Blinking Cursor OFF.
826        when stDisplayCtrlSet =>
827           RS <= LCD_CMDS(lcd_cmd_ptr)(9);
828           RW <= LCD_CMDS(lcd_cmd_ptr)(8);
829           DB <= LCD_CMDS(lcd_cmd_ptr)(7 downto 0);
830           activateW <= '1';
831           stNext <= stDisplayCtrlSet_Delay;
832
833  -- Gives the proper delay of 37us between the display control
     set
834  -- and the Display Clear command.
835        when stDisplayCtrlSet_Delay =>
836           RS <= LCD_CMDS(lcd_cmd_ptr)(9);
837           RW <= LCD_CMDS(lcd_cmd_ptr)(8);
838           DB <= LCD_CMDS(lcd_cmd_ptr)(7 downto 0);
839           activateW <= '0';
840           if delayOK = '1' then
841              stNext <= stDisplayClear;
842           else
843              stNext <= stDisplayCtrlSet_Delay;
844           end if;
```

```
845
846  -- Issues the display clear command.
847        when stDisplayClear       =>
848           RS <= LCD_CMDS(lcd_cmd_ptr)(9);
849           RW <= LCD_CMDS(lcd_cmd_ptr)(8);
850           DB <= LCD_CMDS(lcd_cmd_ptr)(7 downto 0);
851           activateW <= '1';
852           stNext <= stDisplayClear_Delay;
853
854  -- Gives the proper delay of 1.52ms between the clear command
855  -- and the state where you are clear to do normal operations.
856        when stDisplayClear_Delay =>
857           RS <= LCD_CMDS(lcd_cmd_ptr)(9);
858           RW <= LCD_CMDS(lcd_cmd_ptr)(8);
859           DB <= LCD_CMDS(lcd_cmd_ptr)(7 downto 0);
860           activateW <= '0';
861           if delayOK = '1' then
862              stNext <= stInitDne;
863           else
864              stNext <= stDisplayClear_Delay;
865           end if;
866
867  -- State for normal operations for displaying characters,
     changing the
868  -- Cursor position etc.
869        when stInitDne =>
870           RS <= LCD_CMDS(lcd_cmd_ptr)(9);
871           RW <= LCD_CMDS(lcd_cmd_ptr)(8);
872           DB <= LCD_CMDS(lcd_cmd_ptr)(7 downto 0);
873           activateW <= '0';
874           stNext <= stActWr;
```

```
875
876         when stActWr =>
877            RS <= LCD_CMDS(lcd_cmd_ptr)(9);
878            RW <= LCD_CMDS(lcd_cmd_ptr)(8);
879            DB <= LCD_CMDS(lcd_cmd_ptr)(7 downto 0);
880            activateW <= '1';
881            stNext <= stCharDelay;
882
883   -- Provides a max delay between instructions.
884         when stCharDelay =>
885            RS <= LCD_CMDS(lcd_cmd_ptr)(9);
886            RW <= LCD_CMDS(lcd_cmd_ptr)(8);
887            DB <= LCD_CMDS(lcd_cmd_ptr)(7 downto 0);
888            activateW <= '0';
889            if delayOK = '1' then
890               stNext <= stInitDne;
891            else
892               stNext <= stCharDelay;
893            end if;
894         end case;
895   end process;
896
897   -- This process runs the write state machine
898   process (oneUSclk, rst)
899   begin
900      if oneUSclk = '1' and oneUSclk'Event then
901         if rst = '1' then
902            stCurW <= stIdle;
903         else
904            stCurW <= stNextW;
905         end if;
```

```
906      end if;
907   end process;
908
909   -- This genearates the sequence of outputs needed to write to
         the LCD screen
910   process (stCurW, activateW)
911   begin
912      case stCurW is
913   -- This sends the address across the bus telling the DIO5 that
         we are
914   -- writing to the LCD, in this configuration the adr_lcd(2)
         controls the
915   -- enable pin on the LCD
916         when stRw =>
917            E <= '0';
918            stNextW <= stEnable;
919   -- This adds another clock onto the wait to make sure data is
         stable on
920   -- the bus before enable goes low.  The lcd has an active
         falling edge
921   -- and will write on the fall of enable
922         when stEnable =>
923            E <= '0';
924            stNextW <= stIdle;
925   -- Waiting for the write command from the instuction state
         machine
926         when stIdle =>
927            E <= '1';
928            if activateW = '1' then
929               stNextW <= stRw;
930            else
```

```
            stNextW <= stIdle;
          end if;
   end case;
end process;

end Behavioral;
```

refresh_lcd.vhd

```
----------------------------------------------------------------------
-- NELSON SUÁREZ MARTÍN                          refresh_lcd.vhd --
----------------------------------------------------------------------
-- Máster en Ingeniería Industrial                                  --
-- P.F.M.:  Diseño e Implementación de un Instrumento Virtual       --
--          para la Adquisición y Generación de Señales.            --
-- Fecha: Septiembre 2017.                                          --
-- Tutor/es: Oswaldo González & Jonas Lüke                          --
-- Universidad de La Laguna                                         --
--                                                                  --
----------------------------------------------------------------------
--                         DESCRIPCIÓN                              --
----------------------------------------------------------------------
-- Realiza un reset cada 0.16 segundos de la pantalla LCD.          --
-- Este tiempo puede ser modificado desde el paquete                --
-- 'parametros.vhd'.                                                --
--                                                                  --
----------------------------------------------------------------------
--                     refresh_lcd INPUTS                           --
----------------------------------------------------------------------
```

```
21 -- > clk: Señal de reloj de la placa a 50 MHz.                --
22 --                                                             --
23 -- > reset: Señal de reseteo del módulo.                      --
24 --                                                             --
25 -----------------------------------------------------------------
26 --                    refresh_lcd OUTPUTS                     --
27 -----------------------------------------------------------------
28 -- > rst: Reset cada 0.16 segundos del display LCD.           --
29 --                                                             --
30 -----------------------------------------------------------------
31
32 library IEEE;
33 use IEEE.STD_LOGIC_1164.ALL;
34 use work.parametros.all;
35
36 entity refresh_lcd is
37    port( clk : in std_logic;
38          reset : in std_logic;
39            rst : out std_logic);
40 end refresh_lcd;
41
42 architecture Behavioral of refresh_lcd is
43
44 begin
45
46 REFRESH_LCD : process(clk, reset)
47    variable refresh : integer range 0 to actualizar_LCD :=
       actualizar_LCD;
48    variable on_off : std_logic := '0';
49 begin
50    if reset = '1' then
```

```
51          refresh := 0;
52          on_off := '0';
53          rst <= '1';
54       elsif clk'event and clk = '1' then
55          if refresh = actualizar_LCD then
56             case on_off is
57                when '0' => rst <= '1';
58                         on_off := '1';
59                when '1' => rst <= '0';
60                         on_off := '0';
61                when others => rst <= '0';
62                         on_off := '0';
63             end case;
64             refresh := 0;
65          else
66             refresh := refresh+1;
67          end if;
68       end if;
69    end process REFRESH_LCD;
70
71    end Behavioral;
```

oscilloscope.vhd

```
1  ----------------------------------------------------------------------
2  -- NELSON SUÁREZ MARTÍN                          oscilloscope.vhd --
3  ----------------------------------------------------------------------
4  -- Máster en Ingeniería Industrial                                  --
5  -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
```

```
6  --          para la Adquisición y Generación de Señales.      --
7  -- Fecha: Septiembre 2017.                                     --
8  -- Tutor/es: Oswaldo González & Jonas Lüke                     --
9  -- Universidad de La Laguna                                    --
10 --                                                            --
11 ----------------------------------------------------------------
12 --                          DESCRIPCIÓN                       --
13 ----------------------------------------------------------------
14 -- Grupo de componentes encargados del tratamiento de datos   --
15 -- enviados por el ADC. Se encarga también de la configuración--
16 -- del PGA.                                                    --
17 --                                                            --
18 ----------------------------------------------------------------
19 --                   oscilloscope COMPONENTS                  --
20 ----------------------------------------------------------------
21 -- > ADQ_MODULE: Realiza la adquisión de datos del ADC para   --
22 --     su posterior envío a la máquina de estados del sistema.--
23 --                                                            --
24 -- > PGA: Máquina de estados que configura el PGA de la placa --
25 --     externa.                                               --
26 --                                                            --
27 ----------------------------------------------------------------
28 --                    oscilloscope INPUTS                     --
29 ----------------------------------------------------------------
30 -- > clk: Señal de reloj del sistema realizado (50MHz).       --
31 --                                                            --
32 -- > reset: Señal de reseteo del módulo.                      --
33 --                                                            --
34 -- > rst_ADC_clk: Reset de la señal de reloj del ADC cuando   --
35 --     existe un cambio de muestreo.                          --
36 --                                                            --
```

```
37 -- > data_ADC: Datos proporcionados por el ADC.                --
38 --                                                             --
39 -- > read_pointer: Marca el ritmo de envío de datos hacia la  --
40 --     máquina de estados del sistema.                         --
41 --                                                             --
42 -- > adq_time: Tiempo de adquisición del ADC.                  --
43 --                                                             --
44 -- > trigger: El software indica el nivel del umbral de       --
45 --     disparo para la visualización de la señal en la         --
46 --     pantalla del oscilospio del software.                   --
47 --                                                             --
48 -- > OVR: El ADC de la placa externa indica a través de este  --
49 --     bit si la señal está saturada.                          --
50 --                                                             --
51 -- > rst_PGA: Reset del PGA cuando existe un cambio de        --
52 --     ganancia.                                               --
53 --                                                             --
54 -- > gain: Ganancia que se le proporciona al PGA.             --
55 --                                                             --
56 -- > D_out: Reenvío que realiza el PGA de la ganancia que le  --
57 --     habíamos proporcionado en última instancia para         --
58 --     confirmar así su correcta recepción.                    --
59 --                                                             --
60 -------------------------------------------------------------------
61 --                   oscilloscope OUTPUTS                      --
62 -------------------------------------------------------------------
63 -- > OVR_led: Led indicador de la saturación del ADC.         --
64 --                                                             --
65 -- > clk_ADC: Reloj del ADC.                                  --
66 --                                                             --
67 -- > osc_data: Datos de 8 bits enviados a la máquina de       --
```

```
68  --     estados del sistema.                                          --
69  --                                                                   --
70  -- > tx: Transmisión de datos del puerto serie RS232.               --
71  --                                                                   --
72  -- > clk_DAC: Reloj del DAC.                                        --
73  --                                                                   --
74  -- > data_DAC: Envío de datos hacia el DAC.                         --
75  --                                                                   --
76  -- > D_in: Datos de entrada al PGA con la nueva ganancia.           --
77  --    - D_in<0-3> -> Ganancia Canal A.                              --
78  --    - D_in<4-7> -> Ganancia Canal B.                              --
79  --                                                                   --
80  -- > SHDN: Señal de reset del PGA.                                  --
81  --                                                                   --
82  -- > load_data: Habilitar/Deshabilitar el envío de una nueva        --
83  --     ganancia al PGA.                                              --
84  --                                                                   --
85  -- > clk_PGA: Reloj del PGA.                                        --
86  --                                                                   --
87  -----------------------------------------------------------------------
88
89  library IEEE;
90  use IEEE.STD_LOGIC_1164.ALL;
91  use work.parametros.all;
92
93  entity oscilloscope is
94     port(  clk : in std_logic;
95           reset : in std_logic;
96  -- ADQ_MODULE
97    rst_ADC_clk : in std_logic;
98       data_ADC : in std_logic_vector(nbits-1 downto 0);
```

```
 read_pointer : in integer range 0 to L_samples;
     adq_time : in natural range 3 to adq_count;
      trigger : in std_logic_vector(7 downto 0);
          OVR : in std_logic;
      OVR_led : out std_logic;
      clk_ADC : out std_logic;
     osc_data : out std_logic_vector(nbits_com-1 downto 0);
-- PGA
      rst_PGA : in std_logic;
         gain : in std_logic_vector(3 downto 0);
        D_out : in std_logic;
         D_in : out std_logic;
         SHDN : out std_logic;
    load_data : out std_logic;
      clk_PGA : out std_logic);
end oscilloscope;

architecture Behavioral of oscilloscope is

component adq_module is
   port(  clk : in std_logic;
         reset : in std_logic;
  rst_ADC_clk : in std_logic;
      data_in : in std_logic_vector(nbits-1 downto 0);
 read_pointer : in integer range 0 to L_samples;
     adq_time : in natural range 3 to adq_count;
      trigger : in std_logic_vector(7 downto 0);
          OVR : in std_logic;
      OVR_led : out std_logic;
      clk_ADC : out std_logic;
     data_out : out std_logic_vector(nbits_com-1 downto 0));
```

```
130 end component adq_module;
131
132 component PGA is
133    port( clk : in std_logic;
134          reset : in std_logic;
135        rst_PGA : in std_logic;
136           gain : in std_logic_vector(3 downto 0);
137          D_out : in std_logic;
138           D_in : out std_logic;
139           SHDN : out std_logic;
140    load_data : out std_logic;
141        clk_PGA : out std_logic);
142 end component PGA;
143
144 begin
145
146 adq_module_unit : adq_module
147    PORT MAP( clk => clk,
148              reset => reset,
149        rst_ADC_clk => rst_ADC_clk,
150            data_in => data_ADC,
151     read_pointer => read_pointer,
152           adq_time => adq_time,
153            trigger => trigger,
154                OVR => OVR,
155            OVR_led => OVR_led,
156            clk_ADC => clk_ADC,
157           data_out => osc_data);
158
159 PGA_unit : PGA
160    PORT MAP( clk => clk,
```

```
161             reset => reset,
162           rst_PGA => rst_PGA,
163              gain => gain,
164             D_out => D_out,
165              D_in => D_in,
166              SHDN => SHDN,
167         load_data => load_data,
168           clk_PGA => clk_PGA);
169
170 end Behavioral;
```

adq_module.vhd

```
1  ----------------------------------------------------------------
2  -- NELSON SUÁREZ MARTÍN                          adq_module.vhd --
3  ----------------------------------------------------------------
4  -- Máster en Ingeniería Industrial                             --
5  -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
6  --          para la Adquisición y Generación de Señales.      --
7  -- Fecha: Septiembre 2017.                                     --
8  -- Tutor/es: Oswaldo González & Jonas Lüke                     --
9  -- Universidad de La Laguna                                    --
10 --                                                            --
11 ----------------------------------------------------------------
12 --                         DESCRIPCIÓN                        --
13 ----------------------------------------------------------------
14 -- Realiza la adquisición de datos del ADC para su posterior    --
15 -- envío a la máquina de estados del sistema.                  --
16 --                                                            --
17 ----------------------------------------------------------------
```

```
18 --                      adq_module INPUTS                          --
19 -----------------------------------------------------------------
20 -- > clk: Señal de reloj del sistema realizado (50MHz).          --
21 --                                                                 --
22 -- > reset: Señal de reseteo del módulo.                         --
23 --                                                                 --
24 -- > rst_ADC_clk: Reset de la señal de reloj del ADC cuando      --
25 --     existe un cambio de muestreo.                              --
26 --                                                                 --
27 -- > data_in: Datos proporcionados por el ADC.                    --
28 --                                                                 --
29 -- > read_pointer: Marca el ritmo de envío de datos hacia la     --
30 --     máquina de estados del sistema.                            --
31 --                                                                 --
32 -- > adq_time: Tiempo de adquisición del ADC.                     --
33 --                                                                 --
34 -- > trigger: El software indica el nivel del umbral de          --
35 --     disparo para la visualización de la señal en la           --
36 --     pantalla del oscilospio del software.                      --
37 --                                                                 --
38 -- > OVR: El ADC de la placa externa indica a través de este     --
39 --     bit si la señal está saturada.                             --
40 --                                                                 --
41 -----------------------------------------------------------------
42 --                      adq_module OUTPUTS                         --
43 -----------------------------------------------------------------
44 -- > OVR_led: Led indicador de la saturación del ADC.            --
45 --                                                                 --
46 -- > clk_ADC: Reloj del ADC.                                      --
47 --                                                                 --
48 -- > data_out: Datos de 8 bits enviados a la máquina de          --
```

```
49 --     estados del sistema.                                          --
50 --                                                                   --
51 -----------------------------------------------------------------------
52
53 library IEEE;
54 use IEEE.STD_LOGIC_1164.ALL;
55 use IEEE.STD_LOGIC_UNSIGNED.ALL;
56 use IEEE.STD_LOGIC_ARITH.ALL;
57 use work.parametros.all;
58
59 entity adq_module is
60    port(  clk : in std_logic;
61         reset : in std_logic;
62   rst_adc_clk : in std_logic;
63       data_in : in std_logic_vector(nbits-1 downto 0);
64  read_pointer : in integer range 0 to L_samples;
65      adq_time : in natural range 3 to adq_count;
66       trigger : in std_logic_vector(7 downto 0);
67           OVR : in std_logic; -- Over-Range indicator.
68       OVR_led : out std_logic;
69       clk_ADC : out std_logic;
70      data_out : out std_logic_vector(nbits_com-1 downto 0));
71 end adq_module;
72
73 architecture Behavioral of adq_module is
74
75 signal clk_ADC_sig : std_logic := '0';
76 signal memo_adq, memo_adq_buffer : memo;
77 signal trigger_value : std_logic_vector(7 downto 0) := (others
      => '0');
78
```

```
79  begin
80
81  ADC_CLOCK : process(clk, reset, rst_adc_clk)
82     variable count : integer range 0 to adq_count := 0;
83     variable clk_signal : std_logic := '0';
84  begin
85     if (reset or rst_adc_clk) = '1' then
86        count := 0;
87        clk_signal := '0';
88        clk_ADC_sig <= '0';
89     elsif clk'event and clk = '1' then
90        if count = adq_time then
91           clk_signal := not clk_signal;
92           count := 0;
93        end if;
94           count := count + 1;
95     end if;
96  clk_ADC_sig <= clk_signal;
97  end process;
98
99  clk_ADC <= clk_ADC_sig;
100
101 ADQ_PROCESS : process(clk_ADC_sig, reset)
102    variable pointer, trigger_pointer : integer range 0 to
          L_samples := 0;
103    variable reset_memo, trigger_state : boolean := false;
104 begin
105    if reset = '1' then
106       pointer := 0;
107       trigger_pointer := L_samples/2;
108       trigger_value <= (others => '0');
```

```
109         trigger_state := false;
110         reset_memo := false;
111      elsif clk_ADC_sig'event and clk_ADC_sig = '1' then
112
113         if (trigger_value /= trigger) and (trigger_state = true)
               then
114            reset_memo := true;
115            trigger_state := false;
116            trigger_value <= trigger;
117            pointer := 0;
118         end if;
119
120         case reset_memo is
121            when true =>
122               if pointer < L_samples then
123                  memo_adq(pointer) <= (others => '0');
124                  memo_adq_buffer(pointer) <= (others => '0');
125                  pointer := pointer+1;
126               else
127                  pointer := 0;
128                  reset_memo := false;
129               end if;
130            when others =>
131               memo_adq(pointer) <= data_in(nbits-1 downto 4);
132               pointer := (pointer + 1) mod L_samples;
133               trigger_pointer := (pointer + L_samples/2) mod
                     L_samples;
134
135               if data_in(nbits-1 downto 4) < trigger then
136                  trigger_state := true;
137               end if;
```

```
138
139             if read_pointer = 0 and trigger_state = true then
140                memo_adq_buffer(trigger_pointer downto 0) <=
                      memo_adq(trigger_pointer downto 0);
141                memo_adq_buffer(L_samples-1 downto
                      trigger_pointer+1) <= memo_adq(L_samples-1
                      downto trigger_pointer+1);
142             end if;
143       end case;
144
145    end if;
146 end process;
147
148 data_out <= memo_adq_buffer(read_pointer) + 2**7;
149
150 -- Encendido de un led de la placa cuando hay saturación a la
151 -- salida del ADC.
152 OVR_PROCESS : process(clk, reset)
153 begin
154    if reset = '1' then
155       OVR_led <= '0';
156    elsif clk'event and clk = '1' then
157       if OVR = '1' and data_in(nbits-1) = '1' then
158          OVR_led <= '1';
159       else
160          OVR_led <= '0';
161       end if;
162    end if;
163 end process;
164
165 end Behavioral;
```

PGA.vhd

```
1  ----------------------------------------------------------------
2  -- NELSON SUÁREZ MARTÍN                                PGA.vhd --
3  ----------------------------------------------------------------
4  -- Máster en Ingeniería Industrial                             --
5  -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
6  --          para la Adquisición y Generación de Señales.      --
7  -- Fecha: Septiembre 2017.                                     --
8  -- Tutor/es: Oswaldo González & Jonas Lüke                     --
9  -- Universidad de La Laguna                                    --
10 --                                                            --
11 ----------------------------------------------------------------
12 --                        DESCRIPCIÓN                         --
13 ----------------------------------------------------------------
14 -- Máquina de estados que configura el PGA de la placa        --
15 -- externa.                                                   --
16 --                                                            --
17 ----------------------------------------------------------------
18 --                        PGA INPUTS                          --
19 ----------------------------------------------------------------
20 -- > clk: Señal de reloj del sistema realizado (50MHz).       --
21 --                                                            --
22 -- > reset: Señal de reseteo del módulo.                      --
23 --                                                            --
24 -- > rst_PGA: Reset del PGA cuando existe un cambio de        --
25 --     ganancia.                                              --
26 --                                                            --
27 -- > gain: Ganancia que se le proporciona al PGA.            --
28 --                                                            --
```

```
29 -- > D_out: Reenvío que realiza el PGA de la ganancia que le  --
30 --     habíamos proporcionado en última instancia para        --
31 --     confirmar así su correcta recepción.                   --
32 --                                                            --
33 ----------------------------------------------------------------
34 --                          PGA OUTPUTS                       --
35 ----------------------------------------------------------------
36 -- > D_in: Datos de entrada al PGA de la placa con la nueva   --
37 --     ganancia.                                              --
38 --    - D_in<0-3> -> Ganancia Canal A.                        --
39 --    - D_in<4-7> -> Ganancia Canal B.                        --
40 --                                                            --
41 -- > SHDN: Señal de reset del PGA.                           --
42 --                                                            --
43 -- > load_data: Habilitar/Deshabilitar el envío de una nueva  --
44 --     ganancia al PGA.                                       --
45 --                                                            --
46 -- > clk_PGA: Reloj del PGA.                                 --
47 --                                                            --
48 ----------------------------------------------------------------
49
50 library IEEE;
51 use IEEE.STD_LOGIC_1164.ALL;
52 use work.parametros.all;
53
54 entity PGA is
55    port( clk : in std_logic;
56        reset : in std_logic;
57      rst_PGA : in std_logic;
58         gain : in std_logic_vector(3 downto 0);
59        D_out : in std_logic;
```

```
60          D_in : out std_logic;
61          SHDN : out std_logic;
62     load_data : out std_logic;
63       clk_PGA : out std_logic);
64  end PGA;
65
66  architecture Behavioral of PGA is
67
68  signal clk_PGA_sig : std_logic := '0';
69
70  begin
71
72  -- Reloj de 3.125MHz para el PGA siempre y cuando clk = 50MHz.
73  PGA_CLOCK : process(clk, reset)
74     variable clk_signal : std_logic := '0';
75     variable count : natural range 0 to clk_PGA_div-1 := 0;
76  begin
77     if reset = '1' then
78        count := 0;
79        clk_signal := '0';
80        clk_PGA_sig <= '0';
81     elsif clk'event and clk = '1' then
82        if count = 0 then
83           clk_signal := not clk_signal;
84        end if;
85        count := (count+1) mod clk_PGA_div;
86     end if;
87  clk_PGA_sig <= clk_signal;
88  end process;
89
90  clk_PGA <= clk_PGA_sig;
```

```
91
92  -- Proceso de envío de ganancia al PGA de la placa externa.
93  PGA_CONFIG : process(clk_PGA_sig, reset, rst_PGA)
94     variable cuenta: natural range 0 to 10 := 0;
95     variable gain_PGA : std_logic_vector(7 downto 0) :=
          "00010001";
96  begin
97     if (reset or rst_PGA) = '1' then
98        cuenta := 0;
99        D_in <= '0';
100       SHDN <= '0';
101       load_data <= '1';
102       gain_PGA := gain&gain;
103    elsif clk_PGA_sig'event and clk_PGA_sig = '0' then
104       case cuenta is
105          when 0 =>   -- Reset del PGA.
106             SHDN <= '1';
107             load_data <= '1';
108             D_in <= '0';
109          when 1 =>   -- Preparar el PGA para el envío de la
                nueva ganancia.
110             SHDN <= '0';
111             load_data <= '1';
112             D_in <= '0';
113          when 2 to 9 => -- Habilitamos la carga y enviamos los
                datos al PGA.
114             SHDN <= '0';
115             load_data <= '0';
116             D_in <= gain_PGA(9-cuenta);
117          when others => -- Deshabilitamos la carga de datos en
                el PGA.
```

```
118             SHDN <= '0';
119             load_data <= '1';
120             D_in <= '0';
121             cuenta := 9;
122       end case;
123       cuenta := cuenta + 1;
124    end if;
125 end process;
126 
127 end Behavioral;
```

parametros.vhd

```
1  ----------------------------------------------------------------
2  -- NELSON SUÁREZ MARTÍN                          parametros.vhd --
3  ----------------------------------------------------------------
4  -- Máster en Ingeniería Industrial                             --
5  -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
6  --          para la Adquisición y Generación de Señales.      --
7  -- Fecha: Septiembre 2017.                                     --
8  -- Tutor/es: Oswaldo González & Jonas Lüke                     --
9  -- Universidad de La Laguna                                    --
10 --                                                            --
11 ----------------------------------------------------------------
12 --                        DESCRIPCIÓN                         --
13 ----------------------------------------------------------------
14 -- Paquete que contiene todas las constantes, tipos, subtipos --
15 -- y funciones empleadas en el desarrolo del PFM.             --
16 --                                                            --
```

```
17 ------------------------------------------------------------------
18
19 library IEEE;
20 use IEEE.STD_LOGIC_1164.all;
21 use IEEE.STD_LOGIC_ARITH.all;
22 use IEEE.MATH_REAL.all;
23
24 package parametros is
25
26 -- Reloj de la placa.
27 constant f_fpga : natural := 50e6;
28
29 -- Tiempo de actualización de la pantalla LCD.
30 constant actualizar_LCD : natural := f_fpga/6;
31
32 -- Tasa de baudios
33 constant tasa_baudios : natural := 38400;
34
35 -- Paridad --> false: sin paridad; true: con paridad impar.
36 constant paridad_uart : boolean := false;
37
38 -- Tamaño de la tabla para la generación de las señales.
39 constant M : natural := 2**12;
40
41 -- Ciclo de trabajo.
42 constant DCycle : natural := 1024;
43
44 -- Número de bits de entrada al DAC y salida del ADC.
45 constant nbits : natural := 12;
46
47 -- Número de bits para la comunicación serie.
```

```
48  constant nbits_com : natural := 8;
49
50  -- Número de bits de envío a la pantalla LCD.
51  constant nbits_lcd : natural := 4;
52
53  -- Número de decimales al truncar.
54  constant ndec : natural := 10;
55
56  -- Número pi.
57  constant Pi : real := 3.1415927;
58
59  -- Incrementos de fase con decimales.
60  constant delta_fi : real := 2.0*Pi/real(M);
61
62  -- Declaración del subtipo palabra de nbits de longitud.
63  subtype palabra is signed (nbits-1 downto 0);
64
65  -- Tabla de palabras.
66  type tabla is array (M-1 downto 0) of palabra;
67
68  -- Número de muestras adquiridas por el ADC.
69  constant L_samples : natural := 2**8;
70
71  -- Declaración del subtipo palabra_memo de nbits_com de longitud
72  -- para las memorias de adquisición de datos y memoria de envío
73  -- a software.
74  subtype palabra_memo is std_logic_vector(nbits_com-1 downto 0);
75
76  -- Tabla de palabras para la memoria de adquisición de datos.
77  type memo is array (L_samples-1 downto 0) of palabra_memo;
78
```

```
79  -- Cuenta máxima para la adquisición de datos del ADC.
80  constant adq_count : natural := 2**16;
81
82  -- Divisor de frecuencia del reloj de la placa para el PGA.
83  constant divisor_clk_PGA : natural := 16;
84  constant clk_PGA_div : natural := divisor_clk_PGA/2;
85
86  -- Estados para la comunicación serie.
87  type states is (WAITING, SEND_TO_PC, RESTART);
88
89  end parametros;
90
91  package body parametros is
92
93  end parametros;
```

real2bit.vhd

```
1   ----------------------------------------------------------------
2   -- NELSON SUÁREZ MARTÍN (Original de Oswaldo González)       --
3   ----------------------------------------------------------------
4   -- Máster en Ingeniería Industrial                           --
5   -- P.F.M.:  Diseño e Implementación de un Instrumento Virtual --
6   --          para la Adquisición y Generación de Señales.     --
7   -- Fecha: Septiembre 2017.                                    --
8   -- Tutor/es: Oswaldo González & Jonas Lüke                   --
9   -- Universidad de La Laguna                                  --
10  --                                                          --
11  ----------------------------------------------------------------
12  --                         DESCRIPCIÓN                      --
```

```
13 ------------------------------------------------------------
14 -- Definición de las tablas de longitud M = 2**12 para la     --
15 -- generación de las señales senoidal, triangular y cuadrada. --
16 --                                                            --
17 -- Así mismo, se definen dos funciones para la adecuación de  --
18 -- estos datos para su envío hacia el DAC.                    --
19 --                                                            --
20 ------------------------------------------------------------
21
22
23 library IEEE;
24 use IEEE.STD_LOGIC_1164.all;
25 use IEEE.STD_LOGIC_ARITH.ALL;
26 use IEEE.MATH_REAL.ALL;
27 use work.parametros.all;
28
29 package real2bit is
30
31 subtype doble is signed(2*nbits-1 downto 0);
32 function truncar (a: real; numdec : natural := ndec; numbits :
      natural := nbits)        return signed;
33 function extraer (a: doble; numdec : natural := ndec; numbits :
      natural := nbits) return signed;
34
35 -- Generar la tabla para la señal senoidal
36 function inicializar_tabla_senoidal return tabla;
37 constant tabla_senoidal : tabla := inicializar_tabla_senoidal;
38
39 -- Generar la tabla para la señal triangular
40 function inicializar_tabla_triangular return tabla;
```

```
41 constant tabla_triangular : tabla :=
      inicializar_tabla_triangular;
42
43 -- Generar la tabla para la señal cuadrada
44 function inicializar_tabla_cuadrada return tabla;
45 constant tabla_cuadrada : tabla := inicializar_tabla_cuadrada;
46
47 end real2bit;
48
49 package body real2bit is
50
51 function truncar (a: real; numdec : natural := ndec; numbits :
      natural := nbits) return signed is
52    variable resultado : signed(numbits-1 downto 0);
53    variable tmp, comp : real := 0.0;
54    variable comp_int, signo : integer := 0;
55 begin
56    tmp := abs(a*(2.0**numdec));
57    if a < 0.0 then
58       signo:= -1;
59    else
60       signo:= 1;
61    end if;
62
63    for I in numbits-2 downto 0 loop
64       comp := comp + 2.0**I;
65       comp_int := comp_int + 2**I;
66       if tmp < comp then
67          comp := comp - 2.0**I;
68          comp_int := comp_int - 2**I;
69       end if;
```

```
70     end loop;
71
72  resultado := conv_signed(comp_int*signo, numbits);
73
74  return resultado;
75
76  end truncar;
77
78
79  function extraer (a: doble; numdec : natural := ndec; numbits :
       natural := nbits) return signed is
80     variable resultado : signed(numbits-1 downto 0);
81  begin
82     resultado := signed(a(numdec+numbits-1 downto numdec));
83
84  return resultado;
85
86  end function extraer;
87
88
89  function inicializar_tabla_senoidal return tabla is
90     variable resultado : tabla;
91  begin
92     for I in 0 to M-1 loop
93        resultado(I) := truncar(2.0*sin(delta_fi*real(I)));
94     end loop;
95
96  return resultado;
97
98  end function inicializar_tabla_senoidal;
99
```

```
100
101 function inicializar_tabla_triangular return tabla is
102    variable resultado : tabla;
103 begin
104
105    for I in 0 to M-1 loop
106       if I < M/2 then -- Rampa de subida
107          resultado(I) := truncar(8.0*(real(I)/real(M))-2.0);
108       else            -- Rampa de bajada
109          resultado(I) := truncar(-8.0*(real(I)/real(M))+6.0) ;
110       end if;
111    end loop;
112
113 return resultado;
114
115 end function inicializar_tabla_triangular;
116
117
118 function inicializar_tabla_cuadrada return tabla is
119    variable resultado : tabla;
120 begin
121
122    for I in 0 to M-1 loop
123       if I < M/2 then
124          resultado(I) := "010000000000";
125       else
126          resultado(I) := "100000000000";
127       end if;
128    end loop;
129
130 return resultado;
```

```
131
132 end function inicializar_tabla_cuadrada;
133
134 end real2bit;
```

SGandO.ucf

```
1  ######################################################################
2  ## NELSON SUÁREZ MARTÍN                                  SGandO.ucf ##
3  ######################################################################
4  ## Máster en Ingeniería Industrial                                  ##
5  ## P.F.M.:  Diseño e Implementación de un Instrumento Virtual ##
6  ##          para la Adquisición y Generación de Señales.      ##
7  ## Fecha: Septiembre 2017.                                          ##
8  ## Tutor/es: Oswaldo González & Jonas Lüke                          ##
9  ## Universidad de La Laguna                                         ##
10 ##                                                                  ##
11 ######################################################################
12 ##                          DESCRIPCIÓN                             ##
13 ######################################################################
14 ## Archivo de restricciones del proyecto realizado.                 ##
15 ##                                                                  ##
16 ######################################################################
17
18 # Reloj de la placa
19 NET "clk" LOC = "E12";
20 #NET "clk" CLOCK_DEDICATED_ROUTE = FALSE;
21
22 # Switchs
```

```
23 NET "freq_mult_sw" LOC = "T9" | IOSTANDARD = LVTTL | PULLDOWN;
24 #NET "sw<2>" LOC = "U8" | IOSTANDARD = LVTTL | PULLDOWN;
25 NET "sel_signal<1>" LOC = "V8" | IOSTANDARD = LVTTL | PULLDOWN;
26 NET "sel_signal<0>" LOC = "U10"| IOSTANDARD = LVTTL | PULLDOWN;
27
28 # Pulsadores
29 NET "rst" LOC = "T15" | IOSTANDARD = LVTTL | PULLDOWN;
30 NET "rst" CLOCK_DEDICATED_ROUTE = FALSE;
31
32 # Interruptor rotativo
33 NET "ROT_A" LOC = "T13" | IOSTANDARD = LVTTL | PULLUP;
34 NET "ROT_B" LOC = "R14" | IOSTANDARD = LVTTL | PULLUP;
35 NET "sel_param" LOC = "R13" | IOSTANDARD = LVTTL | PULLDOWN;
36 NET "sel_param" CLOCK_DEDICATED_ROUTE = FALSE;
37
38 # Indicadores LED
39 NET "ctrl_led"  LOC = "W21";
40 NET "OVR_led"   LOC = "Y22";
41 #NET "led<5>"   LOC = "V20";
42 #NET "led<4>"   LOC = "V19";
43 NET "sg_led<3>" LOC = "U19";
44 NET "sg_led<2>" LOC = "U20";
45 NET "sg_led<1>" LOC = "T19";
46 NET "sg_led<0>" LOC = "R20";
47
48 # UART
49 NET "rx" LOC = E16; # Recepción de datos
50 NET "tx" LOC = F15; # Transmisión de datos
51
52 # Display LCD
53 NET "DB<0>" LOC = Y13;
```

```
54 NET "DB<1>" LOC = AB18;
55 NET "DB<2>" LOC = AB17;
56 NET "DB<3>" LOC = AB12;
57 NET "DB<4>" LOC = AA12;
58 NET "DB<5>" LOC = Y16;
59 NET "DB<6>" LOC = AB16;
60 NET "DB<7>" LOC = Y15;
61 NET "RS" LOC = Y14;
62 NET "RW" LOC = W13;
63 NET "E" LOC = AB4;
64
65 # FX2 Connector (FX2) --> DAC
66 NET "clk_DAC" LOC = "A13";
67 NET "data_DAC<11>" LOC = "B13";
68 NET "data_DAC<10>" LOC = "A14";
69 NET "data_DAC<9>"  LOC = "B15";
70 NET "data_DAC<8>"  LOC = "A15";
71 NET "data_DAC<7>"  LOC = "A16";
72 NET "data_DAC<6>"  LOC = "A17";
73 NET "data_DAC<5>"  LOC = "B17";
74 NET "data_DAC<4>"  LOC = "A18";
75 NET "data_DAC<3>"  LOC = "C18";
76 NET "data_DAC<2>"  LOC = "A19";
77 NET "data_DAC<1>"  LOC = "B19";
78 NET "data_DAC<0>"  LOC = "A20";
79
80 # FX2 Connector (FX2) --> PGA
81 NET "D_out"     LOC = "D20";
82 NET "SHDN"      LOC = "D21";
83 NET "load_data" LOC = "D22";
84 NET "D_in"      LOC = "E22";
```

```
85  NET "clk_PGA"   LOC = "F18";
86
87  # FX2 Connector (FX2) --> ADC
88  NET "OVR" LOC = "G20";
89  NET "data_ADC<11>" LOC = "G19";
90  NET "data_ADC<10>" LOC = "H19";
91  NET "data_ADC<9>"  LOC = "J18";
92  NET "data_ADC<8>"  LOC = "K18";
93  NET "data_ADC<7>"  LOC = "K17";
94  NET "data_ADC<6>"  LOC = "K19";
95  NET "data_ADC<5>"  LOC = "K20";
96  NET "data_ADC<4>"  LOC = "L19";
97  NET "data_ADC<3>"  LOC = "L18";
98  NET "data_ADC<2>"  LOC = "M20";
99  NET "data_ADC<1>"  LOC = "M18";
100 NET "data_ADC<0>"  LOC = "L20";
101 NET "clk_ADC" LOC = "P20";
```

Anexo C

Códigos C++

mainwindow.h

```
/*****************************************************************
* NELSON SUÁREZ MARTÍN                                mainwindow.h *
*****************************************************************
* Máster en Ingeniería Industrial                                 *
* P.F.M.: Diseño e Implementación de un Instrumento Virtual       *
*         para la Adquisición y Generación de Señales.            *
* Fecha: Septiembre 2017.                                         *
* Tutor/es: Oswaldo González & Jonas Lüke.                        *
* Universidad de La Laguna                                        *
*****************************************************************
* Descripción del fichero:                                        *
*      Cabecera de la clase "mainwindow".                         *
*****************************************************************
* Fuente:                                                         *
*      Elaboración propia.                                        *
*****************************************************************/

```

```
18 #ifndef MAINWINDOW_H
19 #define MAINWINDOW_H
20
21 #include <QMainWindow>
22
23 // API para la comunicación serie
24 #include <QtSerialPort/QSerialPort>
25
26 #include "rs232serialport.h"
27 #include <QVector>
28
29 namespace Ui {
30 class MainWindow;
31 }
32
33 class MainWindow : public QMainWindow
34 {
35     Q_OBJECT
36
37 public:
38     explicit MainWindow(QWidget *parent = 0);
39     ~MainWindow();
40
41     unsigned char b;
42
43 // Variables Generador de Señales.
44     float time_scale;
45     float volt_scale;
46     float volt_act;
47     float offset_act;
48
```

```
49 // Variables Osciloscopio.
50
51
52 public slots:
53    void data_in(QByteArray d);
54    void send_command(unsigned char b);
55
56 private slots:
57 // TABS:
58    void on_actionPuerto_triggered();
59    void on_actionConectar_triggered();
60
61    void on_actionDisconnect_triggered();
62    void on_actionPC_triggered();
63    void on_actionBoard_triggered();
64    void on_actionSine_triggered();
65    void on_actionTriangular_triggered();
66    void on_actionSquare_triggered();
67    void on_actionx1_triggered();
68    void on_actionx100_triggered();
69
70    void on_actionPlay_triggered();
71    void on_actionSave_Picture_triggered();
72    void on_actionDC_triggered();
73    void on_actionAC_triggered();
74
75    void on_actionQt_Webpage_triggered();
76    void on_actionAbout_triggered();
77
78 // BUTTONS:
79    void on_pushButton_RS232_clicked(bool checked);
```

```
80      void on_pushButton_onoff_clicked(bool checked);
81      void on_pushButton_save_clicked();
82      void on_pushButton_play_clicked(bool checked);
83      void on_pushButton_pause_clicked(bool checked);
84      void on_pushButton_sg_clicked(bool checked);
85      void on_pushButton_sine_clicked(bool checked);
86      void on_pushButton_square_clicked(bool checked);
87      void on_pushButton_trian_clicked(bool checked);
88      void on_pushButton_dialFreq1_clicked(bool checked);
89      void on_pushButton_dialFreq100_clicked(bool checked);
90      void on_pushButton_ACDC_clicked(bool checked);
91
92  // DIALS:
93      void on_dial_trigger_valueChanged();
94      void on_dial_volt_sg_valueChanged();
95      void on_dial_freq_valueChanged();
96      void on_dial_offset_valueChanged();
97      void on_dial_dutyCycle_valueChanged();
98      void on_dial_volt_osc_valueChanged();
99      void on_dial_time_valueChanged();
100
101 private:
102     void arranque();
103     void enable_all();
104     void disable_all();
105     void config_port();
106
107 Ui::MainWindow *ui;
108
109 // Parámetros solicitados por el puerto RS232.
110     QString puerto;
```

```
111     QSerialPort rs232;
112     rs232serialPort *port;
113
114 };
115
116 #endif // MAINWINDOW_H
```

mainwindow.cpp

```
1  /*****************************************************************
2  * NELSON SUÁREZ MARTÍN                              mainwindow.cpp *
3  *****************************************************************
4  * Máster en Ingeniería Industrial                                 *
5  * P.F.M.: Diseño e Implementación de un Instrumento Virtual       *
6  *         para la Adquisición y Generación de Señales.            *
7  * Fecha: Septiembre 2017.                                         *
8  * Tutor/es: Oswaldo González & Jonas Lüke.                        *
9  * Universidad de La Laguna.                                       *
10 *****************************************************************
11 * Descripción del fichero:                                        *
12 *      Archivo fuente de la clase "mainwindow". Es donde se       *
13 *      lleva a cabo la programación de todas las instrucciones    *
14 *      que deberán ejecutarse para el funcionamiento del          *
15 *      Generador de Señales y el Osciloscopio. A lo largo del     *
16 *      código se comentan las partes más importantes que se       *
17 *      han realizado.                                             *
18 *****************************************************************
19 * Fuente:                                                         *
20 *      Elaboración propia.                                        *
21 *****************************************************************/
```

```
22
23 #include "mainwindow.h"
24 #include "ui_mainwindow.h"
25 #include "parametros.h"
26 #include "rs232serialport.h"
27 #include "portWindow.h"
28 #include <qcustomplot.h>
29 #include <QMessageBox>
30 #include <QDebug>
31 #include <QDir>
32 #include <QFileDialog>
33 #include <QVector>
34 #include <QUrl>
35
36 /*************************************************************
37 * Ejecución e inicialización del programa.                   *
38 *************************************************************/
39 MainWindow::MainWindow(QWidget *parent) :
40    QMainWindow(parent),
41    ui(new Ui::MainWindow){
42    ui->setupUi(this);
43
44 // Configuración inicial del puerto serie.
45    on_actionPuerto_triggered();
46    port = 0;
47
48 // Establecer valores iniciales.
49 volt_act = 0.25;
50 offset_act = 0.;
51 time_scale = 70.0;
52 volt_scale = 1.0;
```

```
53
54      ui->menuSignal_Generator->setEnabled(false);
55      ui->menuOscilloscope->setEnabled(false);
56      ui->actionConectar->setText("Disconnect RS-232");
57      ui->actionDisconnect->setText("Disconnect");
58      ui->actionConectar->setIcon(QIcon(":/Images/x_icon.png"));
59      ui->actionPC->setEnabled(false);
60      ui->actionBoard->setEnabled(true);
61      ui->actionPlay->setText("Play");
62      ui->actionAC->setEnabled(true);
63      ui->actionDC->setEnabled(false);
64
65      ui->pushButton_onoff->setChecked(true);
66      ui->pushButton_dialFreq1->setChecked(true);
67      ui->pushButton_sine->setChecked(true);
68      ui->pushButton_ACDC->setChecked(false);
69
70      ui->time_scale_label->setText(QString("100 ms"));
71      ui->volt_scale_label->setText(QString("%1 volts").arg(
           volt_scale));
72
73  // Crear gráfica.
74      ui->graphDisplay->addGraph();
75      ui->graphDisplay->axisRect()->setAutoMargins(QCP::msNone);
76      ui->graphDisplay->axisRect()->setMargins(QMargins(0,0,0,0));
77
78  // Inicializar eje 'x'.
79      ui->graphDisplay->xAxis->setRange(0,10*time_scale);
80      ui->graphDisplay->xAxis->setAutoTickStep(true);
81
82  // Inicializar eje 'y'.
```

```
83      ui->graphDisplay->yAxis->setRange(0,5);
84      ui->graphDisplay->yAxis->setAutoTickStep(true);
85
86      qDebug() << "Inicialización correcta";
87
88      arranque();
89  }
90
91  MainWindow::~MainWindow(){
92      delete ui;
93  }
94
95  /*************************************************************
96  * Una vez se han inicializado los diferentes parámetros de    *
97  * SGandO (Signal Generator and Oscilloscope) se procede a     *
98  * la conexión del puerto serie RS232 que hayamos seleccionado *
99  * en la primera ventana "Selección de Puerto Serie".          *
100 *                                                             *
101 * Así mismo, esta llamada la realiza el botón denominado      *
102 * "pushButton_RS232" para la conexión y desconexión del       *
103 * puerto. Esta última acción, la desconexión, inhabilitará    *
104 * todos los botones y diales de SGandO. La acción contraria   *
105 * se lleva a cabo cuando lo que se realiza es la conexión     *
106 * nuevamente del puerto.                                      *
107 *************************************************************/
108 void MainWindow::arranque(){
109
110     if(puerto.size()==0){
111         QMessageBox::critical(this,"Puerto no conectado",QString("
              Conecte un dispositivo de comunicación RS-232 y
              selecciónelo en el menú Configuración."));
```

```
112         ui->pushButton_RS232->setChecked(false);
113         disable_all();
114     }
115     else{
116         ui->pushButton_RS232->setChecked(true);
117         b = 3;
118         send_command(b);
119         enable_all();
120     }
121 }
122
123 /*************************************************************
124 * Habilitación de todos los mandos en caso de conexión       *
125 * correcta con el puerto serie RS232.                        *
126 *************************************************************/
127 void MainWindow::enable_all(){
128     ui->pushButton_onoff->setEnabled(true);
129     ui->pushButton_onoff->setChecked(true);
130     ui->pushButton_save->setEnabled(true);
131     ui->pushButton_play->setEnabled(true);
132     ui->pushButton_pause->setEnabled(true);
133     ui->pushButton_sg->setEnabled(true);
134     ui->groupBoxOsc->setEnabled(true);
135     ui->groupBoxSG->setEnabled(true);
136     on_dial_trigger_valueChanged();
137     ui->actionConectar->setText("Disconnect RS-232");
138     ui->actionConectar->setIcon(QIcon(":/Images/x_icon.png"));
139     ui->actionDisconnect->setText("Disconnect");
140     ui->actionDisconnect->setIcon(QIcon(":/Images/x_icon.png"));
141     ui->menuSignal_Generator->setEnabled(true);
142     ui->menuSG_Control->setEnabled(true);
```

```
143     ui->actionPC->setEnabled(false);
144     ui->actionBoard->setEnabled(true);
145     ui->pushButton_sg->setChecked(false);
146     ui->menuShape->setEnabled(true);
147     ui->menuFreq_multiplier->setEnabled(true);
148     ui->menuOscilloscope->setEnabled(true);
149     ui->actionPlay->setText("Play");
150     ui->dial_freq->setSliderPosition(1);
151     ui->dial_volt_sg->setSliderPosition(1);
152     ui->dial_offset->setSliderPosition(0);
153     ui->dial_dutyCycle->setSliderPosition(4);
154     ui->dial_volt_osc->setSliderPosition(1);
155     ui->dial_time->setSliderPosition(0);
156     ui->dial_trigger->setSliderPosition(5);
157 }
158
159 /*************************************************************
160 * Deshabilitación de todos los mandos en caso de conexión     *
161 * incorrecta o desconexión del puerto serie RS232.            *
162 *************************************************************/
163 void MainWindow::disable_all(){
164     ui->pushButton_onoff->setEnabled(false);
165     ui->pushButton_save->setEnabled(false);
166     ui->pushButton_play->setEnabled(false);
167     ui->pushButton_pause->setEnabled(false);
168     ui->pushButton_sg->setEnabled(false);
169     ui->groupBoxOsc->setEnabled(false);
170     ui->groupBoxSG->setEnabled(false);
171     ui->actionConectar->setText("Connect RS-232");
172     ui->actionConectar->setIcon(QIcon(":/Images/connect_icon.svg
        "));
```

```
173
174     ui->menuSignal_Generator->setEnabled(false);
175     ui->menuOscilloscope->setEnabled(false);
176 }
177
178 /*************************************************************
179 * Configuración del puerto serie RS232.                      *
180 *************************************************************/
181 void MainWindow::config_port(){
182     rs232.setBaudRate(baudios);
183     rs232.setPortName(puerto);
184     rs232.setParity(QSerialPort::NoParity);
185     rs232.setStopBits(QSerialPort::OneStop);
186     rs232.setDataBits(QSerialPort::Data8);
187     rs232.setFlowControl(QSerialPort::NoFlowControl);
188     rs232.open(QIODevice::ReadWrite);
189     qDebug() << QString("Puerto %1 abierto").arg(puerto);
190 }
191
192 /*************************************************************
193 * Apertura de la ventana de diálogo 'portwindow' desde el    *
194 * menú de la ventana principal para la selección de un nuevo *
195 * puerto.                                                    *
196 *************************************************************/
197 void MainWindow::on_actionPuerto_triggered(){
198     PortWindow *pw = new PortWindow(this);
199     if(pw->exec() == QDialog::Accepted){
200         puerto = pw->getPuerto();
201         config_port();
202         arranque();
203     }
```

```
204 }
205
206 /*************************************************************
207 * Abrir o cerrar el puerto serie desde la aplicación. En caso *
208 * de apertura se habilitarán todos los mandos y, en caso de   *
209 * cierre, se deshabilitarán todos.                            *
210 *************************************************************/
211 void MainWindow::on_actionConectar_triggered(){
212    if(ui->pushButton_RS232->isChecked()){
213       ui->actionConectar->setText("Connect RS-232");
214       ui->actionConectar->setIcon(QIcon(":/Images/connect_icon.
          svg"));
215       ui->pushButton_RS232->setChecked(false);
216       on_pushButton_RS232_clicked(false);
217    }
218    else{
219       ui->actionConectar->setText("Disconnect RS-232");
220       ui->actionConectar->setIcon(QIcon(":/Images/x_icon.png"));
221       ui->pushButton_RS232->setChecked(true);
222       on_pushButton_RS232_clicked(true);
223    }
224 }
225
226 void MainWindow::on_pushButton_RS232_clicked(bool checked){
227    if(checked){
228       if(!rs232.open(QIODevice::ReadWrite)){
229          ui->pushButton_RS232->setChecked(false);
230          QMessageBox::critical(this,"Fallo puerto",QString("
             Fallo al abrir el puerto %1. Por favor, seleccione un
              puerto    disponible.").arg(puerto));
231          disable_all();
```

```
232             on_actionPuerto_triggered();
233             return;
234         }
235         else{
236             config_port();
237             enable_all();
238         }
239     }
240     else{
241         rs232.close();
242         qDebug() << QString("Puerto %1 cerrado").arg(puerto);
243         disable_all();
244     }
245 }
246 
247 /*************************************************************
248 * Envío de parámetros hacia la FPGA.                         *
249 *************************************************************/
250 void MainWindow::send_command(unsigned char b){
251     QByteArray ba;
252     ba[0] = b;
253     rs232.write(ba);
254     rs232.flush();
255 }
256 
257 
258 /*************************************************************
259 *                    GENERADOR DE SEÑALES                    *
260 *************************************************************/
261 
262 /*************************************************************
```

```
263 * Habilitar/Deshabilitar Generador de señales.                *
264 ************************************************************/
265 void MainWindow::on_actionDisconnect_triggered(){
266     if(ui->pushButton_onoff->isChecked()){
267         ui->dial_freq->setSliderPosition(1);
268         ui->dial_volt_sg->setSliderPosition(1);
269         ui->dial_offset->setSliderPosition(0);
270         ui->dial_dutyCycle->setSliderPosition(4);
271         ui->dial_volt_osc->setSliderPosition(1);
272         ui->dial_time->setSliderPosition(0);
273         ui->dial_trigger->setSliderPosition(5);
274         ui->actionDisconnect->setText("Connect");
275         ui->pushButton_onoff->setChecked(false);
276         on_pushButton_onoff_clicked(false);
277     }
278     else{
279         ui->dial_freq->setSliderPosition(1);
280         ui->dial_volt_sg->setSliderPosition(1);
281         ui->dial_offset->setSliderPosition(0);
282         ui->dial_dutyCycle->setSliderPosition(4);
283         ui->dial_volt_osc->setSliderPosition(1);
284         ui->dial_time->setSliderPosition(0);
285         ui->dial_trigger->setSliderPosition(5);
286         ui->actionDisconnect->setText("Disconnect");
287         ui->pushButton_onoff->setChecked(true);
288         on_pushButton_onoff_clicked(true);
289     }
290 }
291
292 void MainWindow::on_pushButton_onoff_clicked(bool checked){
293
```

```
294     if(checked){
295         b = 31;
296         ui->pushButton_sg->setEnabled(true);
297         ui->menuShape->setEnabled(true);
298         ui->actionDisconnect->setText("Disconnect");
299         ui->actionDisconnect->setIcon(QIcon(":/Images/x_icon.png")
                );
300         ui->menuSG_Control->setEnabled(true);
301         ui->menuFreq_multiplier->setEnabled(true);
302         on_pushButton_sg_clicked(false);
303     }
304     else {
305         b = 0;
306         ui->pushButton_sg->setEnabled(false);
307         ui->actionDisconnect->setText("Connect");
308         ui->actionDisconnect->setIcon(QIcon(":/Images/connect_icon
                .svg"));
309         ui->menuSG_Control->setEnabled(false);
310         ui->menuFreq_multiplier->setEnabled(false);
311         on_pushButton_sg_clicked(true);
312     }
313
314     qDebug() << "ON/OFF Generador de señales:" << bin << uint8_t(
            b);
315     send_command(b);
316 }
317
318 /*************************************************************
319 * El generador de señales es controlado desde el PC o desde   *
320 * la placa.                                                   *
321 *************************************************************/
```

```
322 void MainWindow::on_actionPC_triggered(){
323     if(ui->pushButton_sg->isChecked()){
324         ui->actionPC->setEnabled(true);
325         ui->actionBoard->setEnabled(false);
326         ui->pushButton_sg->setChecked(false);
327         on_pushButton_sg_clicked(false);
328     }
329     else{
330         on_pushButton_sg_clicked(true);
331     }
332 }
333
334 void MainWindow::on_actionBoard_triggered(){
335     if(ui->pushButton_sg->isChecked()){
336         on_pushButton_sg_clicked(false);
337     }
338     else{
339         ui->actionPC->setEnabled(false);
340         ui->actionBoard->setEnabled(true);
341         ui->pushButton_sg->setChecked(true);
342         on_pushButton_sg_clicked(true);
343     }
344 }
345
346 void MainWindow::on_pushButton_sg_clicked(bool checked){
347
348     ui->dial_freq->setSliderPosition(1);
349     ui->dial_volt_sg->setSliderPosition(1);
350     ui->dial_offset->setSliderPosition(0);
351     ui->dial_dutyCycle->setSliderPosition(4);
352     ui->dial_volt_osc->setSliderPosition(1);
```

```
353     ui->dial_time->setSliderPosition(0);
354     ui->dial_trigger->setSliderPosition(5);
355
356     if(checked){
357        b = 2;
358        ui->groupBoxSG->setEnabled(false);
359        ui->actionBoard->setEnabled(false);
360        ui->menuShape->setEnabled(false);
361        ui->menuFreq_multiplier->setEnabled(false);
362        if(ui->pushButton_onoff->isChecked()){
363           ui->actionPC->setEnabled(true);
364        }
365        else{
366           ui->actionPC->setEnabled(false);
367        }
368        qDebug() << "Generador de señales controlado desde la
              placa: " << bin << uint8_t(b);
369     }
370     else {
371        b = 3;
372        ui->actionPC->setEnabled(false);
373        ui->actionBoard->setEnabled(true);
374        if(ui->pushButton_sg->isChecked()){
375           ui->groupBoxSG->setEnabled(false);
376           ui->actionPC->setEnabled(true);
377           ui->actionBoard->setEnabled(false);
378           ui->menuShape->setEnabled(false);
379           ui->menuFreq_multiplier->setEnabled(false);
380        }
381        else{
382           ui->groupBoxSG->setEnabled(true);
```

```
383             ui->menuShape->setEnabled(true);
384             ui->menuFreq_multiplier->setEnabled(true);
385         }
386         qDebug() << "Generador de señales controlado desde el PC:
                " << bin << uint8_t(b);
387     }
388     send_command(b);
389 }
390
391 /*************************************************************
392 * Seleccionar señal senoidal.                                *
393 *************************************************************/
394 void MainWindow::on_actionSine_triggered(){
395     on_pushButton_sine_clicked(true);
396 }
397
398 void MainWindow::on_pushButton_sine_clicked(bool checked)
399 {
400     ui->pushButton_sine->setChecked(true);
401     if(checked){
402         ui->pushButton_square->setChecked(false);
403         ui->pushButton_trian->setChecked(false);
404         b = 10;
405         qDebug() << "Generación de señal senoidal";
406         send_command(b);
407     }
408 }
409
410 /*************************************************************
411 * Seleccionar señal triangular.                              *
412 *************************************************************/
```

```
413 void MainWindow::on_actionTriangular_triggered(){
414     on_pushButton_trian_clicked(true);
415 }
416 
417 void MainWindow::on_pushButton_trian_clicked(bool checked){
418     ui->pushButton_trian->setChecked(true);
419     if(checked){
420         ui->pushButton_sine->setChecked(false);
421         ui->pushButton_square->setChecked(false);
422         b = 11;
423         qDebug() << "Generación de señal triangular";
424         send_command(b);
425     }
426 }
427 
428 /*************************************************************
429 * Seleccionar señal cuadrada.                                *
430 *************************************************************/
431 void MainWindow::on_actionSquare_triggered(){
432     on_pushButton_square_clicked(true);
433 }
434 
435 void MainWindow::on_pushButton_square_clicked(bool checked){
436     ui->pushButton_square->setChecked(true);
437     if(checked){
438         ui->pushButton_sine->setChecked(false);
439         ui->pushButton_trian->setChecked(false);
440         b = 12;
441         qDebug() << "Generación de señal cuadrada";
442         send_command(b);
443     }
```

```
444 }
445
446 /*************************************************************
447 * Multiplicador de frecuencia x1.                            *
448 *************************************************************/
449 void MainWindow::on_actionx1_triggered(){
450    on_pushButton_dialFreq1_clicked(true);
451 }
452
453 void MainWindow::on_pushButton_dialFreq1_clicked(bool checked){
454    ui->pushButton_dialFreq1->setChecked(true);
455    if(checked){
456       b = 4;
457       ui->pushButton_dialFreq100->setChecked(false);
458       qDebug() << "Sin multiplicador de frecuencia: " << bin <<
          uint8_t(b);
459    }
460    send_command(b);
461 }
462
463 /*************************************************************
464 * Multiplicador de frecuencia x100.                          *
465 *************************************************************/
466 void MainWindow::on_actionx100_triggered(){
467    on_pushButton_dialFreq100_clicked(true);
468 }
469
470 void MainWindow::on_pushButton_dialFreq100_clicked(bool checked)
    {
471    ui->pushButton_dialFreq100->setChecked(true);
472    if(checked){
```

```
473         b = 5;
474         ui->pushButton_dialFreq1->setChecked(false);
475         qDebug() << "Con multiplicador de frecuencia: " << bin <<
                uint8_t(b);
476     }
477     send_command(b);
478 }
479
480 /*************************************************************
481 * Selector de frecuencia de la señal a generar.              *
482 *************************************************************/
483 void MainWindow::on_dial_freq_valueChanged(){
484     if (ui->pushButton_dialFreq1->isChecked()){
485         switch (ui->dial_freq->value()) {
486         case 1:
487             b = 32+2;
488             break;
489         case 2:
490             b = 32+3;
491             break;
492         case 3:
493             b = 32+5;
494             break;
495         case 4:
496             b = 32+11;
497             break;
498         case 5:
499             b = 32+12;
500             break;
501         case 6:
502             b = 32+13;
```

```
503             break;
504         case 7:
505             b = 32+14;
506             break;
507         case 8:
508             b = 32+19;
509             break;
510         case 9:
511             b = 32+20;
512             break;
513         default:
514             QMessageBox::information(this,"Frecuencia máxima",
                    QString("Para generar señales a mayores frecuencias,
                    por favor, cambie la escala."));
515             ui->dial_freq->setSliderPosition(9);
516         break;
517         }
518     }
519     else{
520         switch (ui->dial_freq->value()) {
521         case 1:
522             b = 32+1;
523             break;
524         case 2:
525             b = 32+2;
526             break;
527         case 3:
528             b = 32+4;
529             break;
530         case 4:
531             b = 32+8;
```

```
532             break;
533         case 5:
534             b = 32+9;
535             break;
536         case 6:
537             b = 32+12;
538             break;
539         case 7:
540             b = 32+13;
541             break;
542         case 8:
543             b = 32+17;
544             break;
545         case 9:
546             b = 6;
547             break;
548         case 10:
549             b = 32+20;
550             break;
551         case 11:
552             b = 32+22;
553             break;
554         case 12:
555             b = 32+26;
556             break;
557         default:
558             b = 32+ui->dial_freq->value();
559             break;
560         }
561     }
562
```

```
563     qDebug() << "Dial frequency: " << bin << uint8_t(b);
564     send_command(b);
565 }
566 
567 /*************************************************************
568 * Selector de voltaje de la señal a generar.                 *
569 *************************************************************/
570 void MainWindow::on_dial_volt_sg_valueChanged(){
571     b = 64+ui->dial_volt_sg->value();
572 
573     switch (ui->dial_volt_sg->value()) {
574     case 1:
575         volt_act = 0.25;
576         break;
577     case 2:
578         volt_act = 0.5;
579         break;
580     case 3:
581         volt_act = 0.7;
582         break;
583     case 4:
584         volt_act = 0.9;
585         break;
586     case 5:
587         volt_act = 1.;
588         break;
589     case 6:
590         volt_act = 1.25;
591         break;
592     case 7:
593         volt_act = 1.5;
```

```
594         break;
595     case 8:
596         volt_act = 1.7;
597         break;
598     case 9:
599         volt_act = 2.;
600         break;
601     default:
602         break;
603 }
604 
605     if((volt_act+offset_act > V_max) || (volt_act+offset_act < -
        V_max)){
606         QMessageBox::warning(this, "Valor fuera de rango", QString
            ("Saturación del generador de señales."));
607         ui->dial_volt_sg->setSliderPosition((ui->dial_volt_sg->
            value())-1);
608         b = 64+ui->dial_volt_sg->value();
609     }
610 
611     qDebug() << "Voltage dial: " << bin << uint8_t(b);
612     send_command(b);
613 }
614 
615 /*************************************************************
616 * Offset de la señal a generar.                              *
617 *************************************************************/
618 void MainWindow::on_dial_offset_valueChanged(){
619     if (ui->pushButton_ACDC->isChecked()){
620         b = 96;
621         ui->dial_offset->setSliderPosition(0);
```

```
622         QMessageBox::information(this, "Modo AC activado", QString
               ("Mientras el modo AC se encuentre activo, no se puede
               actuar sobre el offset."));
623     }
624     else{
625         if(ui->dial_offset->value() > 0){
626             b = -ui->dial_offset->value()-384;
627             offset_act = (ui->dial_offset->value())/20.;
628             if(volt_act+2*offset_act > V_max){
629                 QMessageBox::warning(this, "Valor fuera de rango",
                       QString("Saturación del generador de señales."));
630                 ui->dial_offset->setSliderPosition((ui->dial_offset
                       ->value())-1);
631                 b = -ui->dial_offset->value()-384;
632             }
633     }
634     else{
635         b = -ui->dial_offset->value()+96;
636         offset_act = (ui->dial_offset->value())/20.;
637         if(volt_act-2*offset_act > V_max){
638             QMessageBox::warning(this, "Valor fuera de rango",
                    QString("Saturación del generador de señales."));
639             ui->dial_offset->setSliderPosition((ui->dial_offset->
                    value())+1);
640             b = -ui->dial_offset->value()+96;
641         }
642     }
643 }
644
645     qDebug() << "Offset: " << bin << uint8_t(b);
646     send_command(b);
```

```
647 }
648
649 /*************************************************************
650 * Ciclo de trabajo de la señal a generar.                    *
651 *************************************************************/
652 void MainWindow::on_dial_dutyCycle_valueChanged(){
653    b = 128+ui->dial_dutyCycle->value();
654    qDebug() << "Con multiplicador de frecuencia: " << bin <<
          uint8_t(b);
655    send_command(b);
656 }
657
658
659 /*************************************************************
660 *                         OSCILOSCOPIO                       *
661 *************************************************************/
662
663 /*************************************************************
664 * Representación gráfica de los datos de entrada.            *
665 *************************************************************/
666 void MainWindow::data_in(QByteArray d){
667
668    QVector<double> x(MEMO_DISPLAY), y(MEMO_DISPLAY);
669
670    for(int i = 0; i < MEMO_DISPLAY; i++){
671       x[i] = i/double(MEMO_DISPLAY-1)*time_scale*10;
672       if (ui->pushButton_ACDC->isChecked()){
673          y[i] = -d[i]/(32.);
674       }
675       else{
676          y[i] = -d[i]/(32.)+2.;
```

```
677         }
678     }
679     ui->graphDisplay->axisRect()->setAutoMargins(QCP::msNone);
680     ui->graphDisplay->axisRect()->setMargins(QMargins(0,0,0,0));
681     ui->graphDisplay->graph(0)->setData(x,y);
682 
683     ui->graphDisplay->xAxis->setRange(0,time_scale*10);
684     ui->graphDisplay->xAxis->setAutoTickStep(true);
685 
686     ui->graphDisplay->yAxis->setRange(-5,5);
687     ui->graphDisplay->yAxis->setAutoTickStep(true);
688 
689     // Earth line
690     ui->graphDisplay->addGraph();
691     QVector<double> earth_line_x(2), earth_line_y(2);
692     for(int i = 0; i < 2; i++){
693         earth_line_x[i] = i*time_scale*10;
694         earth_line_y[i] = 0;
695     }
696 
697     ui->graphDisplay->graph(1)->setData(earth_line_x,earth_line_y
        );
698     ui->graphDisplay->graph(1)->setPen(QPen(QColor(0,0,0),2));
699 
700     // Trigger
701     ui->graphDisplay->addGraph();
702     QVector<double> trig_x(2), trig_y(2);
703     for(int i = 0; i < 2; i++){
704         trig_x[i] = i*time_scale*10;
705         switch (ui->dial_trigger->value()) {
706         case 0:
```

```
707             trig_y[i] = -7.5;
708             break;
709          case 1:
710             trig_y[i] = -3.;
711             break;
712          case 2:
713             trig_y[i] = -2.25;
714             break;
715          case 3:
716             trig_y[i] = -1.5;
717             break;
718          case 4:
719             trig_y[i] = -0.75;
720             break;
721          case 5:
722             trig_y[i] = 0;
723             break;
724          case 6:
725             trig_y[i] = 0.75;
726             break;
727          case 7:
728             trig_y[i] = 1.5;
729             break;
730          case 8:
731             trig_y[i] = 2.25;
732             break;
733          case 9:
734             trig_y[i] = 3.;
735             break;
736          case 10:
737             trig_y[i] = 7.5;
```

```
738             break;
739         default:
740             break;
741         }
742     }
743 
744     ui->graphDisplay->graph(2)->setData(trig_x,trig_y);
745     ui->graphDisplay->graph(2)->setPen(QPen(QColor(255,0,0),1,Qt
        ::DotLine));
746 
747     ui->graphDisplay->replot();
748 }
749 
750 /*************************************************************
751 * Dibujar puntos de entrada por el puerto serie.             *
752 *************************************************************/
753 void MainWindow::on_actionPlay_triggered(){
754     if(ui->pushButton_play->isChecked()){
755         on_pushButton_pause_clicked(true);
756     }
757     else{
758         on_pushButton_play_clicked(true);
759     }
760 }
761 
762 void MainWindow::on_pushButton_play_clicked(bool checked){
763 
764     ui->pushButton_play->setChecked(true);
765 
766     ui->graphDisplay->axisRect()->setAutoMargins(QCP::msNone);
767     ui->graphDisplay->axisRect()->setMargins(QMargins(0,0,0,0));
```

```
768     ui->graphDisplay->setInteraction(QCP::iRangeDrag, false);
769     ui->graphDisplay->setInteraction(QCP::iRangeZoom, false);
770 
771     if(checked){
772        ui->pushButton_pause->setChecked(false);
773        ui->actionPlay->setText("Pause");
774        ui->actionPlay->setIcon(QIcon(":/Images/pause_icon.png"));
775        b = 4;
776        qDebug() << "Recibir datos: " << bin << uint8_t(b);
777        send_command(b);
778        config_port();
779        if(port != 0){
780           delete port;
781           port = 0;
782        }
783 
784        port = new rs232serialPort(&rs232);
785        connect(port, SIGNAL(dataReady(QByteArray)), this, SLOT(
              data_in(QByteArray)));
786        port->restart();
787     }
788 
789 }
790 
791 /*************************************************************
792 * Parar la representación gráfica.                           *
793 *************************************************************/
794 void MainWindow::on_pushButton_pause_clicked(bool checked){
795 
796    ui->pushButton_pause->setChecked(true);
797 
```

```
798     disconnect(port, SIGNAL(dataReady(QByteArray)), this, SLOT(
            data_in(QByteArray)));
799
800     ui->graphDisplay->setInteraction(QCP::iRangeDrag, true);
801     ui->graphDisplay->setInteraction(QCP::iRangeZoom, true);
802
803     if(checked){
804         ui->pushButton_play->setChecked(false);
805         ui->actionPlay->setText("Play");
806         ui->actionPlay->setIcon(QIcon(":/Images/play_icon.png"));
807         qDebug() << "NO SE RECIBEN DATOS";
808     }
809 }
810
811 /*************************************************************
812 * Guardar la pantalla actual.                                *
813 *************************************************************/
814 void MainWindow::on_actionSave_Picture_triggered(){
815     on_pushButton_save_clicked();
816 }
817
818 void MainWindow::on_pushButton_save_clicked(){
819
820     if (ui->pushButton_pause->isChecked()){
821         QString fileName = QFileDialog::getSaveFileName(this,"
                Guardar gráfica como...");
822         ui->graphDisplay->savePng(fileName+".png",800,600,1.0,-1);
823     }
824     else{
825         QMessageBox::information(this,"Osciloscopio en modo RUN",
                QString("El osciloscopio se encuentra actualizando datos
```

```
          . Presione PAUSE si quiere guardar la gráfica."));
826    }
827 }
828
829 /*************************************************************
830 * Cambio en la escala de voltaje.                           *
831 *************************************************************/
832 void MainWindow::on_dial_volt_osc_valueChanged(){
833    switch (ui->dial_volt_osc->value()){
834    case 1:
835       volt_scale = 1.;
836       break;
837    case 2:
838       volt_scale = 0.5;
839       break;
840    case 3:
841       volt_scale = 0.25;
842       break;
843    case 4:
844       volt_scale = 0.125;
845       break;
846    case 5:
847       volt_scale = 0.0625;
848       break;
849    case 6:
850       volt_scale = 0.03125;
851       break;
852    case 7:
853       volt_scale = 0.015625;
854       break;
855    default:
```

```
856         break;
857     }
858 
859     b = 224+ui->dial_volt_osc->value();
860     ui->volt_scale_label->setText(QString("%1 volts").arg(
            volt_scale));
861     qDebug() << "Ganancia del PGA: " << bin << uint8_t(b);
862     send_command(b);
863 }
864 
865 /*************************************************************
866 * Cambio en la escala de tiempo.                             *
867 *************************************************************/
868 void MainWindow::on_dial_time_valueChanged(){
869     switch (ui->dial_time->value()){
870     case 0:
871         time_scale = 70.;
872         ui->time_scale_label->setText(QString("100 ms"));
873         break;
874     case 1:
875         time_scale = 35.;
876         ui->time_scale_label->setText(QString("50 ms"));
877         break;
878     case 2:
879         time_scale = 17.5;
880         ui->time_scale_label->setText(QString("25 ms"));
881         break;
882     case 3:
883         time_scale = 8.75;
884         ui->time_scale_label->setText(QString("10 ms"));
885         break;
```

```
886     case 4:
887        time_scale = 4.375;
888        ui->time_scale_label->setText(QString("5 ms"));
889        break;
890     case 5:
891        time_scale = 2.2;
892        ui->time_scale_label->setText(QString("2.5 ms"));
893        break;
894     case 6:
895        time_scale = 1.1;
896        ui->time_scale_label->setText(QString("1.25 ms"));
897        break;
898     case 7:
899        time_scale = 0.8;
900        ui->time_scale_label->setText(QString("1 ms"));
901        break;
902     case 8:
903        time_scale = 0.6;
904        ui->time_scale_label->setText(QString("0.8 ms"));
905        break;
906     case 9:
907        time_scale = 0.4;
908        ui->time_scale_label->setText(QString("0.6 ms"));
909        break;
910     case 10:
911        time_scale = 0.25;
912        ui->time_scale_label->setText(QString("0.4 ms"));
913        break;
914     case 11:
915        time_scale = 0.125;
916        ui->time_scale_label->setText(QString("0.2 ms"));
```

```
917         break;
918     case 12:
919         time_scale = 0.065;
920         ui->time_scale_label->setText(QString("0.1 ms"));
921         break;
922     case 13:
923         time_scale = 32.5/1000.;
924         ui->time_scale_label->setText(QString("50 us"));
925         break;
926     case 14:
927         time_scale = 16./1000.;
928         ui->time_scale_label->setText(QString("25 us"));
929         break;
930     case 15:
931         time_scale = 15./1000.;
932         ui->time_scale_label->setText(QString("20 us"));
933         break;
934     case 16:
935         time_scale = 10./1000.;
936         ui->time_scale_label->setText(QString("15 us"));
937         break;
938     case 17:
939         time_scale = 5./1000.;
940         ui->time_scale_label->setText(QString("10 us"));
941         break;
942     case 18:
943         time_scale = 4./1000.;
944         ui->time_scale_label->setText(QString("6 us"));
945         break;
946     case 19:
947         time_scale = 3./1000.;
```

```
948         ui->time_scale_label->setText(QString("4 us"));
949         break;
950     default:
951         break;
952     }
953 
954     b = 160+ui->dial_time->value();
955     qDebug() << "Escala de tiempos: " << bin << uint8_t(b);
956     send_command(b);
957 }
958 
959 /*************************************************************
960 * Selector de umbral de disparo.                             *
961 *************************************************************/
962 void MainWindow::on_dial_trigger_valueChanged(){
963     switch (ui->dial_trigger->value()){
964     case 0:
965         b = 222;
966         break;
967     case 1:
968         b = 218;
969         break;
970     case 2:
971         b = 212;
972         break;
973     case 3:
974         b = 210;
975         break;
976     case 4:
977         b = 209;
978         break;
```

```
979     case 5:
980        b = 208;
981        break;
982     case 6:
983        b = 206;
984        break;
985     case 7:
986        b = 205;
987        break;
988     case 8:
989        b = 203;
990        break;
991     case 9:
992        b = 197;
993        break;
994     case 10:
995        b = 193;
996        break;
997     default:
998        break;
999     }
1000
1001    qDebug() << "Trigger: " << bin << uint8_t(b);
1002    send_command(b);
1003 }
1004
1005 /*************************************************************
1006 * Modo AC/DC.                                                *
1007 *************************************************************/
1008 void MainWindow::on_actionDC_triggered(){
1009    on_pushButton_ACDC_clicked(false);
```

```
1010 }
1011
1012 void MainWindow::on_actionAC_triggered(){
1013    on_pushButton_ACDC_clicked(true);
1014 }
1015
1016 void MainWindow::on_pushButton_ACDC_clicked(bool checked){
1017    if(checked){
1018       ui->pushButton_ACDC->setChecked(true);
1019       ui->actionAC->setEnabled(false);
1020       ui->actionDC->setEnabled(true);
1021       on_dial_offset_valueChanged();
1022       qDebug() << "Modo AC activado";
1023    }
1024    else{
1025       ui->pushButton_ACDC->setChecked(false);
1026       ui->actionAC->setEnabled(true);
1027       ui->actionDC->setEnabled(false);
1028       qDebug() << "Modo AC desactivado";
1029    }
1030 }
1031
1032 /*************************************************************
1033 * Pestaña 'About'.                                           *
1034 *************************************************************/
1035 void MainWindow::on_actionQt_Webpage_triggered(){
1036    QDesktopServices::openUrl(QUrl("https://www.qt.io/",QUrl::
          TolerantMode));
1037 }
1038
1039 void MainWindow::on_actionAbout_triggered(){
```

```
1040     QMessageBox::information(this, "About", QString(
1041 "*****************************************************************\
     n"
1042 "*                                NELSON SUÁREZ MARTÍN
                                   *\n"
1043 "*****************************************************************\
     n"
1044 "***********           Máster en Ingeniería Industrial
     **********\n"
1045 "*****************************************************************\
     n"
1046 "*                             UNIVERSIDAD DE LA LAGUNA
                               *\n"
1047 "*****************************************************************\
     n\n"
1048 "         P.F.M.: Diseño e Implementación de un Instrumento
                 \n"
1049 "          Virtual para la Adquisición y Generación de Señales.
                \n\n"
1050 "                                        Septiembre 2017
                                         \n\n"
1051 "                Tutor/es: Oswaldo González & Jonas Lüke
                        \n\n"
1052 "*****************************************************************")
     );
1053
1054 }
```

portwindow.h

```
/*****************************************************************
* NELSON SUÁREZ MARTÍN                                portwindow.h *
*****************************************************************
* Máster en Ingeniería Industrial                                *
* P.F.M.: Diseño e Implementación de un Instrumento Virtual      *
*         para la Adquisición y Generación de Señales.           *
* Fecha: Septiembre 2017.                                        *
* Tutor/es: Oswaldo González & Jonas Lüke.                       *
* Universidad de La Laguna.                                      *
*****************************************************************
* Descripción del fichero:                                       *
*      Cabecera de la clase "portwindow".                        *
*****************************************************************
* Fuente:                                                        *
*      Elaboración propia.                                       *
*****************************************************************/

#ifndef PORTWINDOW_H
#define PORTWINDOW_H

#include <QDialog>
#include <QString>
#include <QtSerialPort/QSerialPort>

namespace Ui {
   class PortWindow;
}

class PortWindow : public QDialog
{
```

```
31      Q_OBJECT
32
33  public:
34      explicit PortWindow(QWidget *parent = 0);
35      ~PortWindow();
36
37      QString getPuerto();
38
39  private:
40      Ui::PortWindow *ui;
41
42  };
43
44  #endif // PORTWINDOW_H
```

portwindow.cpp

```
1   /*************************************************************
2   * NELSON SUÁREZ MARTÍN                          portwindow.cpp *
3   *************************************************************
4   * Máster en Ingeniería Industrial                             *
5   * P.F.M.: Diseño e Implementación de un Instrumento Virtual   *
6   *         para la Adquisición y Generación de Señales.        *
7   * Fecha: Septiembre 2017.                                     *
8   * Tutor/es: Oswaldo González & Jonas Lüke.                    *
9   * Universidad de La Laguna.                                   *
10  *************************************************************
11  * Descripción del fichero:                                    *
12  *      Archivo fuente de la clase "portwindow". Crea la       *
13  *      ventana inicial para la selección del puerto serie de  *
```

```
14  *      comunicación antes del inicio del programa principal.   *
15  ************************************************************
16  * Fuente:                                                     *
17  *      Elaboración propia.                                    *
18  ************************************************************/
19
20  #include "portwindow.h"
21  #include "ui_portwindow.h"
22  #include <QtSerialPort/QSerialPortInfo>
23
24  PortWindow::PortWindow(QWidget *parent) :
25  QDialog(parent),
26  ui(new Ui::PortWindow)
27  {
28      ui->setupUi(this);
29
30      for(QSerialPortInfo puerto : QSerialPortInfo::availablePorts
            ()){
31          QString port = puerto.portName();
32          ui->cboPuerto->addItem(port);
33      }
34  }
35
36  PortWindow::~PortWindow(){
37      delete ui;
38  }
39
40  QString PortWindow::getPuerto(){
41      return ui->cboPuerto->currentText();
42  }
```

rs232serialport.h

```
/*************************************************************
* NELSON SUÁREZ MARTÍN                          rs232serialport.h *
*************************************************************
* Máster en Ingeniería Industrial                              *
* P.F.M.: Diseño e Implementación de un Instrumento Virtual    *
*         para la Adquisición y Generación de Señales.         *
* Fecha: Septiembre 2017.                                      *
* Tutor/es: Oswaldo González & Jonas Lüke.                     *
* Universidad de La Laguna.                                    *
*************************************************************
* Descripción del fichero:                                     *
*      Cabecera de la clase "rs232serialport".                 *
*************************************************************
* Fuente:                                                      *
*      Qt Documentation -> Qt Serial Port.                     *
*      http://doc.qt.io/qt-5/qtserialport-index.html           *
*************************************************************/

#ifndef RS232SERIALPORT_H
#define RS232SERIALPORT_H

#include <QtSerialPort/QSerialPort>

#include <QTextStream>
#include <QTimer>
#include <QByteArray>
#include <QObject>

```

```
29 QT_USE_NAMESPACE
30
31 class rs232serialPort : public QObject
32 {
33     Q_OBJECT
34
35 public:
36     explicit rs232serialPort(QSerialPort *serialPort, QObject *
           parent = nullptr);
37     ~rs232serialPort();
38
39     void restart();
40
41 signals:
42     void dataReady(QByteArray d);
43
44 private slots:
45     void handleReadyRead();
46     void handleTimeout();
47     void handleError(QSerialPort::SerialPortError error);
48
49 private:
50     QSerialPort *m_serialPort;
51     QByteArray m_readData;
52     QTextStream m_standardOutput;
53     QTimer m_timer;
54
55     QByteArray memoria;
56
57     int k;
58 };
```

```
59
60 #endif // SERIALPORT_H
```

rs232serialport.cpp

```
1  /*************************************************************
2  * NELSON SUÁREZ MARTÍN                      rs232serialport.cpp *
3  *************************************************************
4  * Máster en Ingeniería Industrial                             *
5  * P.F.M.: Diseño e Implementación de un Instrumento Virtual   *
6  *         para la Adquisición y Generación de Señales.        *
7  * Fecha: Septiembre 2017.                                     *
8  * Tutor/es: Oswaldo González & Jonas Lüke.                    *
9  * Universidad de La Laguna.                                   *
10 *************************************************************
11 * Descripción del fichero:                                    *
12 *      Archivo fuente de la clase "rs232serialport".          *
13 *************************************************************
14 * Fuente:                                                     *
15 *      Qt Documentation -> Qt Serial Port.                    *
16 *      http://doc.qt.io/qt-5/qtserialport-index.html          *
17 *************************************************************/
18
19 #include "rs232serialport.h"
20 #include <QDebug>
21 #include <QCoreApplication>
22
23 #include "parametros.h"
24
```

```
25 QT_USE_NAMESPACE
26
27 /* Inicialización y conexiones.                                    */
28 rs232serialPort::rs232serialPort(QSerialPort *serialPort,
      QObject *parent)
29 :QObject(parent)
30 , m_serialPort(serialPort)
31 , m_standardOutput(stdout)
32 , memoria(MEMO_DISPLAY,0)
33 {
34     connect(m_serialPort, &QSerialPort::readyRead, this, &
         rs232serialPort::handleReadyRead);
35     connect(m_serialPort, static_cast<void (QSerialPort::*)(
         QSerialPort::SerialPortError)>(&QSerialPort::error), this,
         &rs232serialPort::handleError);
36     connect(&m_timer, &QTimer::timeout, this, &rs232serialPort::
         handleTimeout);
37
38     m_timer.start(2000);
39
40     k = 0;
41 }
42
43 rs232serialPort::~rs232serialPort(){
44
45 }
46
47 /* Inicialización y puesta a 0 de la memoria de datos de       *
48 * entrada.                                                     */
49 void rs232serialPort::restart(){
50     k = 0;
```

```
51     for(int i = 0; i < memoria.size(); i++){
52        memoria[i] = 0;
53        m_readData[i] = 0;
54     }
55  }
56
57  /* Lectura de los datos del puerto serie.                    */
58  void rs232serialPort::handleReadyRead(){
59
60     m_readData = m_serialPort->readAll();
61
62     for(int i = 0; i < m_readData.size(); i++){
63        if (k < MEMO_DISPLAY){
64           memoria[k] = m_readData[i];
65           k++;
66           if (k == MEMO_DISPLAY){
67              QByteArray c = memoria;
68              emit dataReady(c);
69              k = 0;
70           }
71        }
72     }
73  }
74
75  /* Estado de la lectura de datos del puerto serie.           */
76  void rs232serialPort::handleTimeout()
77  {
78     if (m_readData.isEmpty()) {
79        m_standardOutput << QObject::tr("No data was currently
           available for reading from port %1").arg(m_serialPort->
           portName()) << endl;
```

```
80        } else {
81        m_standardOutput << QObject::tr("Data successfully
             received from port %1").arg(m_serialPort->portName()) <<
              endl;
82        m_standardOutput << m_readData << endl;
83     }
84  }
85
86  /* Errores en el puerto serie.                                  */
87  void rs232serialPort::handleError(QSerialPort::SerialPortError
       serialPortError){
88     if(serialPortError == QSerialPort::ReadError){
89        qDebug() << QObject::tr("An I/O error occurred while
             reading the data from port %1, error: %2").arg(
             m_serialPort->portName()).arg(m_serialPort->errorString
             ())<<endl;
90        QCoreApplication::exit(1);
91     }
92  }
```

Anexo D

Objetos en *Inkscape*

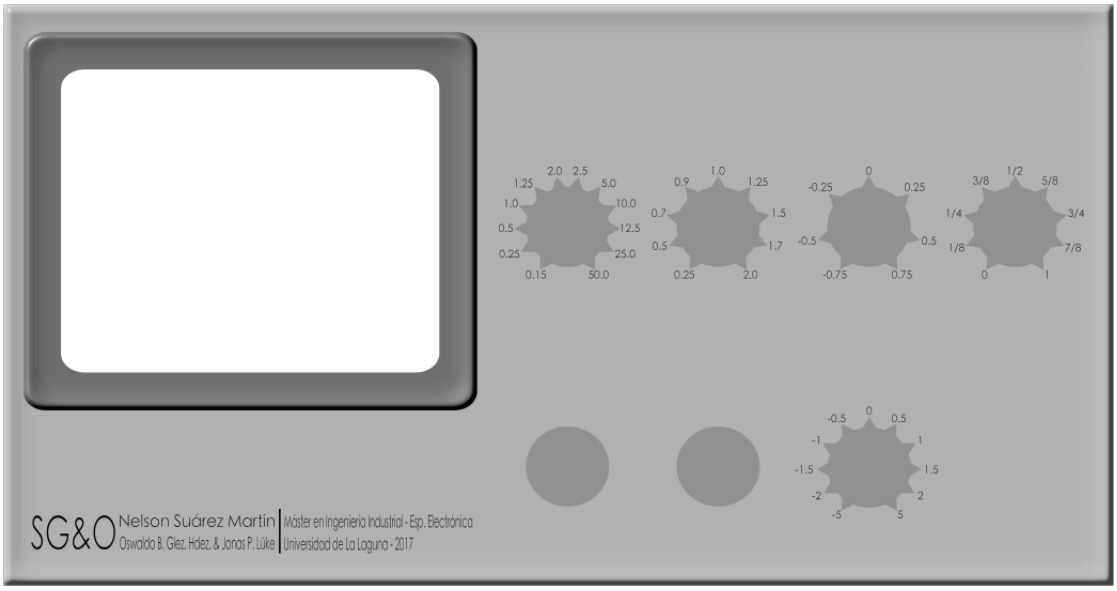


Figura D.1: *Background* de la aplicación SGandO (Escala 1:2).

Figura D.2: Estados del interruptor para la conexión y desconexión del puerto RS–232 (Escala 1:1).

Figura D.3: Estados del interruptor para el apagado y encendido del generador de señales (Escala 1:1).

Figura D.4: Estados del interruptor para guardar gráfica actual (Escala 1:1).

Figura D.5: Estados del interruptor para adquirir datos (Escala 1:1).

Figura D.6: Estados del interruptor para congelar los datos actuales (Escala 1:1).

Figura D.7: Estados del selector de control por medio de software o placa (Escala 1:1).

Figura D.8: Estados del selector para cambio de modo AC–DC (Escala 1:1).

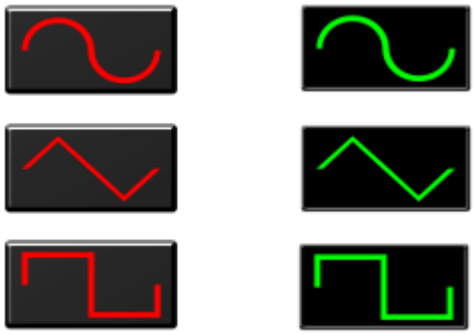


Figura D.9: Estados de los selectores de señal a generar (Escala 1:1).

Figura D.10: Estados de los interruptores multiplicadores de frecuencia (Escala 1:1).

Figura D.11: Iconos utilizados en el menú de SGandO (Escala 1:1).

Anexo E

Hojas de características de componentes

12-Bit, 125 MSPS High Performance TxDAC® D/A Converter

AD9752*

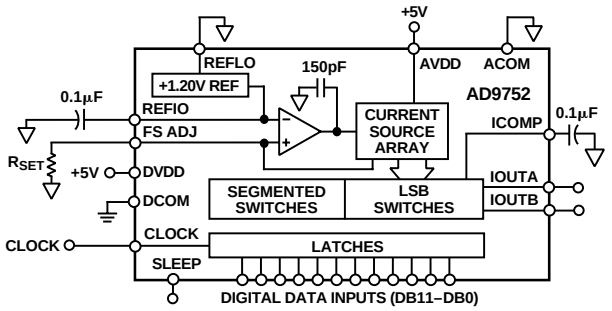
FEATURES

High Performance Member of Pin-Compatible TxDAC Product Family
125 MSPS Update Rate
12-Bit Resolution
Excellent Spurious Free Dynamic Range Performance
SFDR to Nyquist @ 5 MHz Output: 79 dBc
Differential Current Outputs: 2 mA to 20 mA
Power Dissipation: 185 mW @ 5 V
Power-Down Mode: 20 mW @ 5 V
On-Chip 1.20 V Reference
CMOS-Compatible +2.7 V to +5.5 V Digital Interface
Package: 28-Lead SOIC and TSSOP
Edge-Triggered Latches

APPLICATIONS

Wideband Communication Transmit Channel:
Direct IF
Basestations
Wireless Local Loop
Digital Radio Link
Direct Digital Synthesis (DDS)
Instrumentation

FUNCTIONAL BLOCK DIAGRAM



The AD9752 is a current-output DAC with a nominal full-scale output current of 20 mA and > 100 kΩ output impedance.

Differential current outputs are provided to support single-ended or differential applications. Matching between the two current outputs ensures enhanced dynamic performance in a differential output configuration. The current outputs may be tied directly to an output resistor to provide two complementary, single-ended voltage outputs or fed directly into a transformer. The output voltage compliance range is 1.25 V.

The on-chip reference and control amplifier are configured for maximum accuracy and flexibility. The AD9752 can be driven by the on-chip reference or by a variety of external reference voltages. The internal control amplifier, which provides a wide (>10:1) adjustment span, allows the AD9752 full-scale current to be adjusted over a 2 mA to 20 mA range while maintaining excellent dynamic performance. Thus, the AD9752 may operate at reduced power levels or be adjusted over a 20 dB range to provide additional gain ranging capabilities.

The AD9752 is available in 28-lead SOIC and TSSOP packages. It is specified for operation over the industrial temperature range.

PRODUCT DESCRIPTION

The AD9752 is a 12-bit resolution, wideband, second generation member of the TxDAC series of high performance, low power CMOS digital-to-analog-converters (DACs). The TxDAC family, which consists of pin compatible 8-, 10-, 12-, and 14-bit DACs, is specifically optimized for the transmit signal path of communication systems. All of the devices share the same interface options, small outline package and pinout, thus providing an upward or downward component selection path based on performance, resolution and cost. The AD9752 offers exceptional ac and dc performance while supporting update rates up to 125 MSPS.

The AD9752's flexible single-supply operating range of 4.5 V to 5.5 V and low power dissipation are well suited for portable and low power applications. Its power dissipation can be further reduced to a mere 65 mW, without a significant degradation in performance, by lowering the full-scale current output. Also, a power-down mode reduces the standby power dissipation to approximately 20 mW.

The AD9752 is manufactured on an advanced CMOS process. A segmented current source architecture is combined with a proprietary switching technique to reduce spurious components and enhance dynamic performance. Edge-triggered input latches and a 1.2 V temperature compensated bandgap reference have been integrated to provide a complete monolithic DAC solution. The digital inputs support +2.7 V to +5 V CMOS logic families.

TxDAC is a registered trademark of Analog Devices, Inc.
*Protected by U.S. Patents Numbers 5450084, 5568145, 5689257, 5612697 and 5703519.

REV. 0

PRODUCT HIGHLIGHTS

1. The AD9752 is a member of the wideband TxDAC product family that provides an upward or downward component selection path based on resolution (8 to 14 bits), performance and cost. The entire family of TxDACs is available in industry standard pinouts.
2. Manufactured on a CMOS process, the AD9752 uses a proprietary switching technique that enhances dynamic performance beyond that previously attainable by higher power/cost bipolar or BiCMOS devices.
3. On-chip, edge-triggered input CMOS latches interface readily to +2.7 V to +5 V CMOS logic families. The AD9752 can support update rates up to 125 MSPS.
4. A flexible single-supply operating range of 4.5 V to 5.5 V and a wide full-scale current adjustment span of 2 mA to 20 mA allow the AD9752 to operate at reduced power levels.
5. The current output(s) of the AD9752 can be easily configured for various single-ended or differential circuit topologies.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329-4700 World Wide Web Site: http://www.analog.com
Fax: 781/326-8703

AD9752* Product Page Quick Links

Last Content Update: 11/01/2016

Comparable Parts

View a parametric search of comparable parts

Evaluation Kits

- AD9752 Evaluation Board

Documentation

Application Notes

- AN-237: Choosing DACs for Direct Digital Synthesis
- AN-320A: CMOS Multiplying DACs and Op Amps Combine to Build Programmable Gain Amplifier, Part 1
- AN-595: Understanding Pin Compatibility in the TxDAC® Line of High Speed D/A Converters
- AN-912: Driving a Center-Tapped Transformer with a Balanced Current-Output DAC

Data Sheet

- AD9752: 12-Bit, 125 MSPS High Performance TxDAC® D/A Converter Data Sheet

Tools and Simulations

- AD9752 IBIS Models

Reference Materials

Informational

- Advantiv™ Advanced TV Solutions

Solutions Bulletins & Brochures

- Digital to Analog Converters ICs Solutions Bulletin

Design Resources

- AD9752 Material Declaration
- PCN-PDN Information
- Quality And Reliability
- Symbols and Footprints

Discussions

View all AD9752 EngineerZone Discussions

Sample and Buy

Visit the product page to see pricing options

Technical Support

Submit a technical question or find your regional support number

AD9752–SPECIFICATIONS

DC SPECIFICATIONS (T_{MIN} to T_{MAX}, AVDD = +5 V, DVDD = +5 V, I_{OUTFS} = 20 mA, unless otherwise noted)

Parameter	Min	Typ	Max	Units
RESOLUTION	12			Bits
DC ACCURACY[1]				
Integral Linearity Error (INL)				
T_A = +25°C	–1.5	±0.5	+1.5	LSB
T_{MIN} to T_{MAX}	–2.0		+2.0	LSB
Differential Nonlinearity (DNL)				
T_A = +25°C	–0.75	±0.25	+0.75	LSB
T_{MIN} to T_{MAX}	–1.0		+1.0	LSB
ANALOG OUTPUT				
Offset Error	–0.02		+0.02	% of FSR
Gain Error (Without Internal Reference)	–2	±0.5	+2	% of FSR
Gain Error (With Internal Reference)	–5	±1.5	+5	% of FSR
Full-Scale Output Current[2]	2.0		20.0	mA
Output Compliance Range	–1.0		1.25	V
Output Resistance		100		kΩ
Output Capacitance		5		pF
REFERENCE OUTPUT				
Reference Voltage	1.14	1.20	1.26	V
Reference Output Current[3]		100		nA
REFERENCE INPUT				
Input Compliance Range	0.1		1.25	V
Reference Input Resistance		1		MΩ
Small Signal Bandwidth		0.5		MHz
TEMPERATURE COEFFICIENTS				
Offset Drift		0		ppm of FSR/°C
Gain Drift (Without Internal Reference)		±50		ppm of FSR/°C
Gain Drift (With Internal Reference)		±100		ppm of FSR/°C
Reference Voltage Drift		±50		ppm/°C
POWER SUPPLY				
Supply Voltages				
AVDD	4.5	5.0	5.5	V
DVDD	2.7	5.0	5.5	V
Analog Supply Current (I_{AVDD})[4]		34	39	mA
Digital Supply Current (I_{DVDD})[5]		3	5	mA
Supply Current Sleep Mode (I_{AVDD})[6]		4	8	mA
Power Dissipation[5] (5 V, I_{OUTFS} = 20 mA)		185	220	mW
Power Supply Rejection Ratio[7]—AVDD	–0.4		+0.4	% of FSR/V
Power Supply Rejection Ratio[7]—DVDD	–0.025		+0.025	% of FSR/V
OPERATING RANGE	–40		+85	°C

NOTES

[1]Measured at IOUTA, driving a virtual ground.

[2]Nominal full-scale current, I_{OUTFS}, is 32 × the I_{REF} current.

[3]Use an external buffer amplifier to drive any external load.

[4]Requires +5 V supply.

[5]Measured at f_{CLOCK} = 25 MSPS and I_{OUT} = static full scale (20 mA).

[6]Logic level for SLEEP pin must be referenced to AVDD. Min V_{IH} = 3.5 V.

[7]±5% Power supply variation.

Specifications subject to change without notice.

DYNAMIC SPECIFICATIONS

(T_{MIN} to T_{MAX}, AVDD = +5 V, DVDD = +5 V, I_{OUTFS} = 20 mA, Differential Transformer Coupled Output, 50 Ω Doubly Terminated, unless otherwise noted)

Parameter	Min	Typ	Max	Units
DYNAMIC PERFORMANCE				
Maximum Output Update Rate (f_{CLOCK})	125			MSPS
Output Settling Time (t_{ST}) (to 0.1%)[1]		35		ns
Output Propagation Delay (t_{PD})		1		ns
Glitch Impulse		5		pV-s
Output Rise Time (10% to 90%)[1]		2.5		ns
Output Fall Time (10% to 90%)[1]		2.5		ns
Output Noise (I_{OUTFS} = 20 mA)		50		pA/$\sqrt{Hz}$
Output Noise (I_{OUTFS} = 2 mA)		30		pA/$\sqrt{Hz}$
AC LINEARITY				
Spurious-Free Dynamic Range to Nyquist				
f_{CLOCK} = 25 MSPS; f_{OUT} = 1.00 MHz				
0 dBFS Output				
T_A = +25°C	75	84		dBc
–6 dBFS Output		76		dBc
–12 dBFS Output		81		dBc
f_{CLOCK} = 50 MSPS; f_{OUT} = 1.00 MHz		81		dBc
f_{CLOCK} = 50 MSPS; f_{OUT} = 2.51 MHz		81		dBc
f_{CLOCK} = 50 MSPS; f_{OUT} = 5.02 MHz		76		dBc
f_{CLOCK} = 50 MSPS; f_{OUT} = 14.02 MHz		62		dBc
f_{CLOCK} = 50 MSPS; f_{OUT} = 20.2 MHz		60		dBc
f_{CLOCK} = 100 MSPS; f_{OUT} = 2.5 MHz		78		dBc
f_{CLOCK} = 100 MSPS; f_{OUT} = 5 MHz		76		dBc
f_{CLOCK} = 100 MSPS; f_{OUT} = 20 MHz		63		dBc
f_{CLOCK} = 100 MSPS; f_{OUT} = 40 MHz		55		dBc
Spurious-Free Dynamic Range within a Window				
f_{CLOCK} = 25 MSPS; f_{OUT} = 1.00 MHz	84	93		dBc
f_{CLOCK} = 50 MSPS; f_{OUT} = 5.02 MHz; 2 MHz Span		86		dBc
f_{CLOCK} = 100 MSPS; f_{OUT} = 5.04 MHz; 4 MHz Span		86		dBc
Total Harmonic Distortion				
f_{CLOCK} = 25 MSPS; f_{OUT} = 1.00 MHz				
T_A = +25°C		–82	–74	dBc
f_{CLOCK} = 50 MHz; f_{OUT} = 2.00 MHz		–76		dBc
f_{CLOCK} = 100 MHz; f_{OUT} = 2.00 MHz		–76		dBc
Multitone Power Ratio (8 Tones at 110 kHz Spacing)				
f_{CLOCK} = 20 MSPS; f_{OUT} = 2.00 MHz to 2.99 MHz				
0 dBFS Output		81		dBc
–6 dBFS Output		81		dBc
–12 dBFS Output		85		dBc
–18 dBFS Output		86		dBc

NOTES

[1]Measured single ended into 50 Ω load.

Specifications subject to change without notice.

AD9752

DIGITAL SPECIFICATIONS (T_{MIN} to T_{MAX}, AVDD = +5 V, DVDD = +5 V, I_{OUTFS} = 20 mA, unless otherwise noted)

Parameter	Min	Typ	Max	Units
DIGITAL INPUTS				
Logic "1" Voltage @ DVDD = +5 V[1]	3.5	5		V
Logic "1" Voltage @ DVDD = +3 V	2.1	3		V
Logic "0" Voltage @ DVDD = +5 V[1]		0	1.3	V
Logic "0" Voltage @ DVDD = +3 V		0	0.9	V
Logic "1" Current	–10		+10	µA
Logic "0" Current	–10		+10	µA
Input Capacitance		5		pF
Input Setup Time (t_S)	2.0			ns
Input Hold Time (t_H)	1.5			ns
Latch Pulsewidth (t_{LPW})	3.5			ns

NOTES
[1]When DVDD = +5 V and Logic 1 voltage ≈3.5 V and Logic 0 voltage ≈1.3 V. IVDD can increase by up to 10 mA, depending on f_{CLOCK}.
Specifications subject to change without notice.

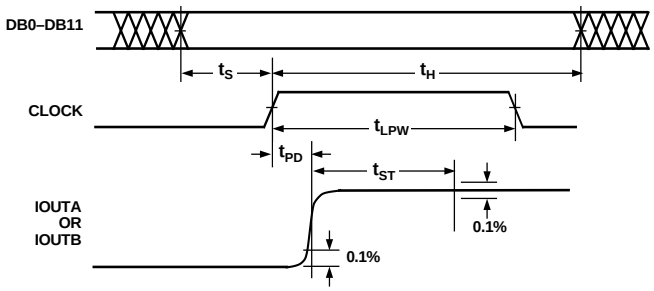


Figure 1. Timing Diagram

ABSOLUTE MAXIMUM RATINGS*

Parameter	With Respect to	Min	Max	Units
AVDD	ACOM	–0.3	+6.5	V
DVDD	DCOM	–0.3	+6.5	V
ACOM	DCOM	–0.3	+0.3	V
AVDD	DVDD	–6.5	+6.5	V
CLOCK, SLEEP	DCOM	–0.3	DVDD + 0.3	V
Digital Inputs	DCOM	–0.3	DVDD + 0.3	V
IOUTA, IOUTB	ACOM	–1.0	AVDD + 0.3	V
ICOMP	ACOM	–0.3	AVDD + 0.3	V
REFIO, FSADJ	ACOM	–0.3	AVDD + 0.3	V
REFLO	ACOM	–0.3	+0.3	V
Junction Temperature			+150	°C
Storage Temperature		–65	+150	°C
Lead Temperature (10 sec)			+300	°C

*Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum ratings for extended periods may effect device reliability.

ORDERING GUIDE

Model	Temperature Range	Package Description	Package Options*
AD9752AR	–40°C to +85°C	28-Lead 300 Mil SOIC	R-28
AD9752ARU	–40°C to +85°C	28-Lead TSSOP	RU-28
AD9752-EB		Evaluation Board	

*R = Small Outline IC; RU = Thin Shrink Small Outline Package.

THERMAL CHARACTERISTICS

Thermal Resistance

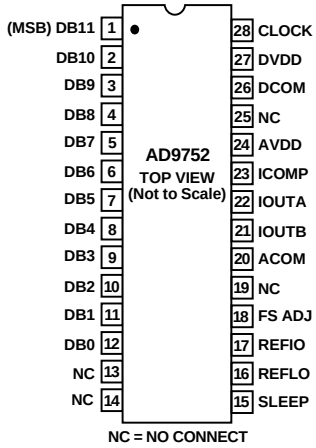
28-Lead 300 Mil SOIC
θ_{JA} = 71.4°C/W
θ_{JC} = 23°C/W

28-Lead TSSOP
θ_{JA} = 97.9°C/W
θ_{JC} = 14.0°C/W

CAUTION

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although the AD9752 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.

PIN CONFIGURATION



PIN FUNCTION DESCRIPTIONS

Pin No.	Name	Description
1	DB11	Most Significant Data Bit (MSB).
2–11	DB10–DB1	Data Bits 1–10.
12	DB0	Least Significant Data Bit (LSB).
13, 14, 19, 25	NC	No Internal Connection.
15	SLEEP	Power-Down Control Input. Active High. Contains active pull-down circuit, thus may be left unterminated if not used.
16	REFLO	Reference Ground when Internal 1.2 V Reference Used. Connect to AVDD to disable internal reference.
17	REFIO	Reference Input/Output. Serves as reference input when internal reference disabled (i.e., Tie REFLO to AVDD). Serves as 1.2 V reference output when internal reference activated (i.e., Tie REFLO to ACOM). Requires 0.1 μF capacitor to ACOM when internal reference activated.
18	FS ADJ	Full-Scale Current Output Adjust.
19	NC	No Connect.
20	ACOM	Analog Common.
21	IOUTB	Complementary DAC Current Output. Full-scale current when all data bits are 0s.
22	IOUTA	DAC Current Output. Full-scale current when all data bits are 1s.
23	ICOMP	Internal Bias Node for Switch Driver Circuitry. Decouple to ACOM with 0.1 μF capacitor.
24	AVDD	Analog Supply Voltage (+4.5 V to +5.5 V).
26	DCOM	Digital Common.
27	DVDD	Digital Supply Voltage (+2.7 V to +5.5 V).
28	CLOCK	Clock Input. Data latched on positive edge of clock.

DEFINITIONS OF SPECIFICATIONS

Linearity Error (Also Called Integral Nonlinearity or INL)
Linearity error is defined as the maximum deviation of the actual analog output from the ideal output, determined by a straight line drawn from zero to full scale.

Differential Nonlinearity (or DNL)
DNL is the measure of the variation in analog value, normalized to full scale, associated with a 1 LSB change in digital input code.

Monotonicity
A D/A converter is monotonic if the output either increases or remains constant as the digital input increases.

Offset Error
The deviation of the output current from the ideal of zero is called offset error. For IOUTA, 0 mA output is expected when the inputs are all 0s. For IOUTB, 0 mA output is expected when all inputs are set to 1s.

Gain Error
The difference between the actual and ideal output span. The actual span is determined by the output when all inputs are set to 1s minus the output when all inputs are set to 0s.

Output Compliance Range
The range of allowable voltage at the output of a current-output DAC. Operation beyond the maximum compliance limits may cause either output stage saturation or breakdown resulting in nonlinear performance.

Temperature Drift
Temperature drift is specified as the maximum change from the ambient (+25°C) value to the value at either T_{MIN} or T_{MAX}. For offset and gain drift, the drift is reported in ppm of full-scale range (FSR) per °C. For reference drift, the drift is reported in ppm per °C.

Power Supply Rejection
The maximum change in the full-scale output as the supplies are varied from nominal to minimum and maximum specified voltages.

Settling Time
The time required for the output to reach and remain within a specified error band about its final value, measured from the start of the output transition.

Glitch Impulse
Asymmetrical switching times in a DAC give rise to undesired output transients that are quantified by a glitch impulse. It is specified as the net area of the glitch in pV-s.

Spurious-Free Dynamic Range
The difference, in dB, between the rms amplitude of the output signal and the peak spurious signal over the specified bandwidth.

Total Harmonic Distortion
THD is the ratio of the rms sum of the first six harmonic components to the rms value of the measured input signal. It is expressed as a percentage or in decibels (dB).

Multitone Power Ratio
The spurious-free dynamic range for an output containing multiple carrier tones of equal amplitude. It is measured as the difference between the rms amplitude of a carrier tone to the peak spurious signal in the region of a removed tone.

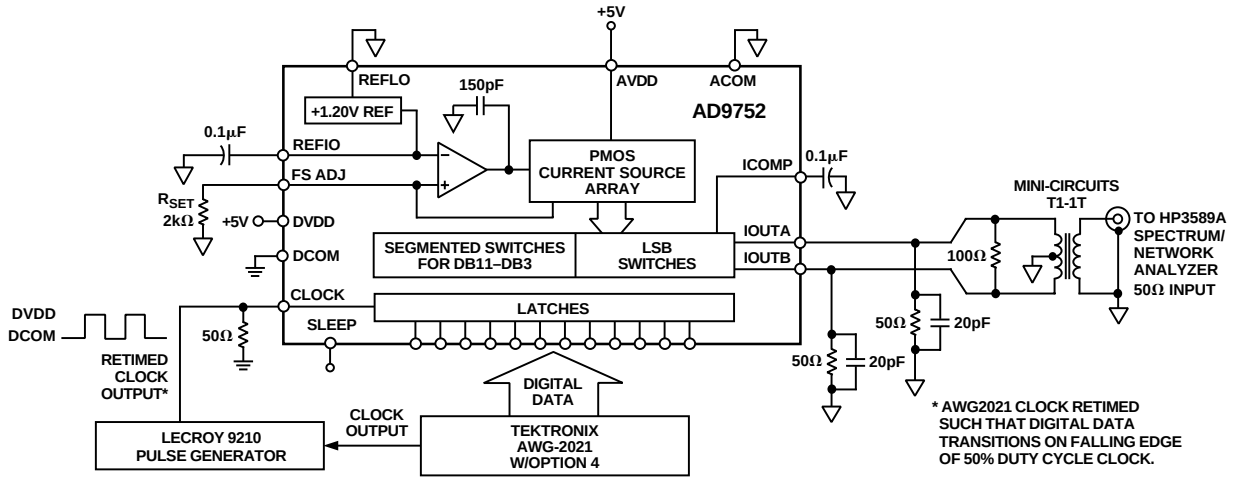


Figure 2. Basic AC Characterization Test Setup

Typical AC Characterization Curves @ +5 V Supplies

(AVDD = +5 V, DVDD = +5 V, I_{OUTFS} = 20 mA, 50 Ω Doubly Terminated Load, Differential Output, T_A = +25°C, SFDR up to Nyquist, unless otherwise noted)

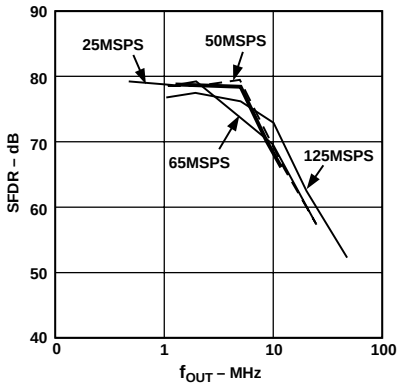


Figure 3. SFDR vs. f_{OUT} @ 0 dBFS

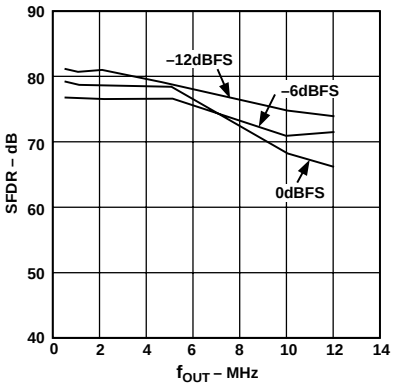


Figure 4. SFDR vs. f_{OUT} @ 25 MSPS

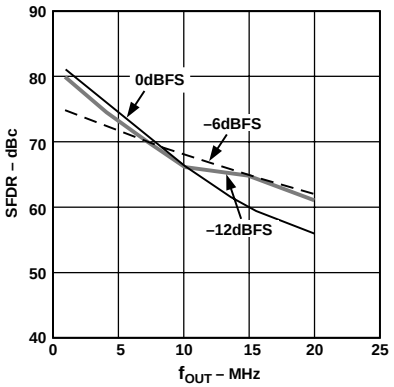


Figure 5. SFDR vs. f_{OUT} @ 50 MSPS

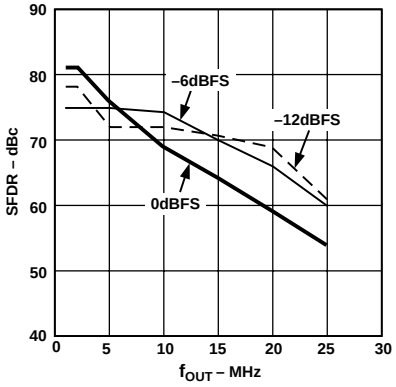


Figure 6. SFDR vs. f_{OUT} @ 65 MSPS

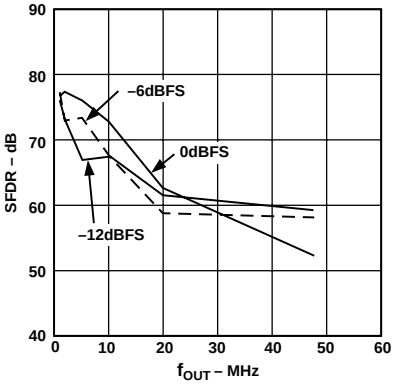


Figure 7. SFDR vs. f_{OUT} @ 125 MSPS

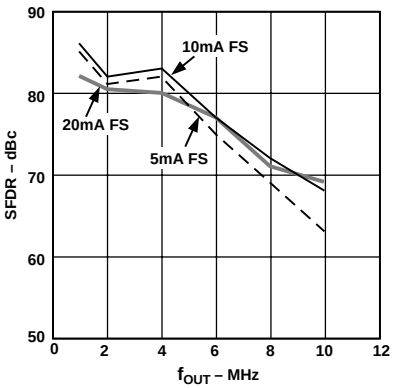


Figure 8. SFDR vs. f_{OUT} and I_{OUTFS} @ 25 MSPS and 0 dBFS

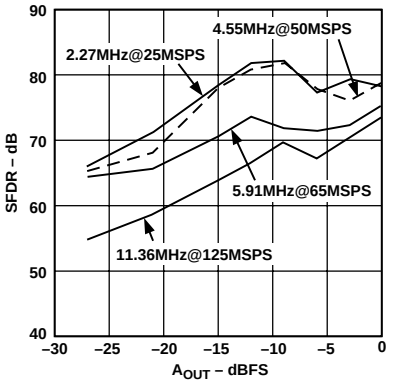


Figure 9. Single-Tone SFDR vs. A_{OUT} @ $f_{OUT} = f_{CLOCK}/11$

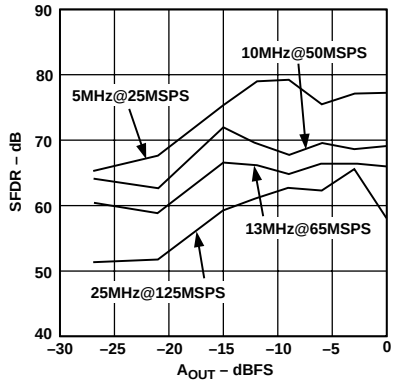


Figure 10. Single-Tone SFDR vs. A_{OUT} @ $f_{OUT} = f_{CLOCK}/5$

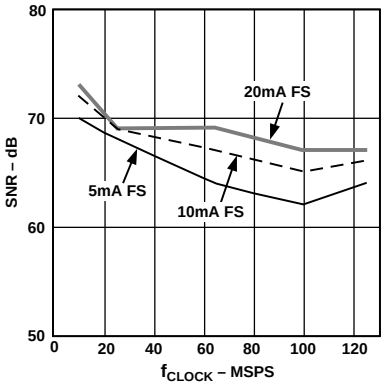


Figure 11. SNR vs. f_{CLOCK} and I_{OUTFS} @ f_{OUT} = 2 MHz and 0 dBFS

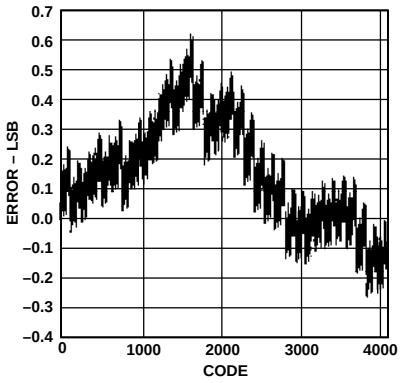


Figure 12. Typical INL

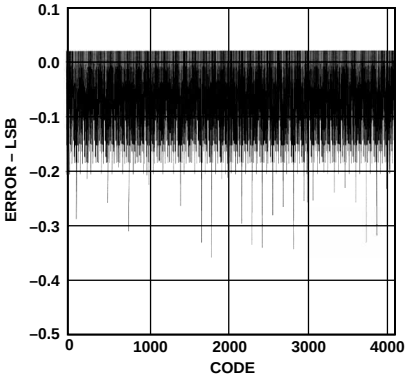


Figure 13. Typical DNL

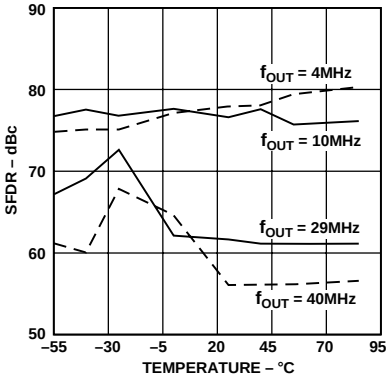


Figure 14. SFDR vs. Temperature @ 125 MSPS, 0 dBFS

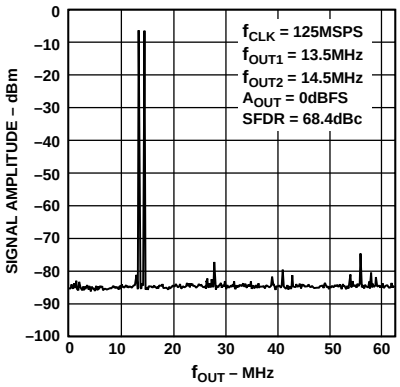


Figure 15. Dual-Tone SFDR

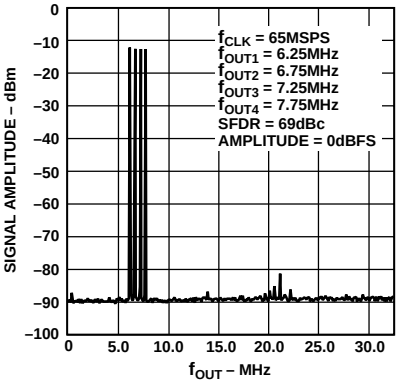


Figure 16. Four-Tone SFDR

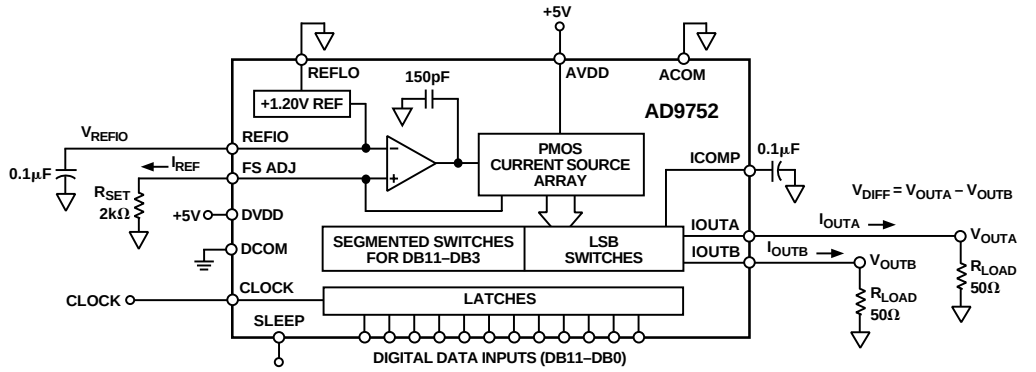


Figure 17. Functional Block Diagram

FUNCTIONAL DESCRIPTION

Figure 17 shows a simplified block diagram of the AD9752. The AD9752 consists of a large PMOS current source array that is capable of providing up to 20 mA of total current. The array is divided into 31 equal currents that make up the five most significant bits (MSBs). The next four bits or middle bits consist of 15 equal current sources whose value is 1/16th of an MSB current source. The remaining LSBs are binary weighted fractions of the middle-bits current sources. Implementing the middle and lower bits with current sources, instead of an R-2R ladder, enhances its dynamic performance for multitone or low amplitude signals and helps maintain the DAC's high output impedance (i.e., >100 kΩ).

All of these current sources are switched to one or the other of the two output nodes (i.e., IOUTA or IOUTB) via PMOS differential current switches. The switches are based on a new architecture that drastically improves distortion performance. This new switch architecture reduces various timing errors and provides matching complementary drive signals to the inputs of the differential current switches.

The analog and digital sections of the AD9752 have separate power supply inputs (i.e., AVDD and DVDD). The digital section, which is capable of operating up to a 125 MSPS clock rate and over a +2.7 V to +5.5 V operating range, consists of edge-triggered latches and segment decoding logic circuitry. The analog section, which can operate over a +4.5 V to +5.5 V range, includes the PMOS current sources, the associated differential switches, a 1.20 V bandgap voltage reference and a reference control amplifier.

The full-scale output current is regulated by the reference control amplifier and can be set from 2 mA to 20 mA via an external resistor, R_{SET}. The external resistor, in combination with both the reference control amplifier and voltage reference V_{REFIO}, sets the reference current I_{REF}, which is mirrored over to the segmented current sources with the proper scaling factor. The full-scale current, I_{OUTFS}, is thirty-two times the value of I_{REF}.

DAC TRANSFER FUNCTION

The AD9752 provides complementary current outputs, IOUTA and IOUTB. IOUTA will provide a near full-scale current output, I_{OUTFS}, when all bits are high (i.e., DAC CODE = 4095) while IOUTB, the complementary output, provides no current. The current output appearing at IOUTA and IOUTB is a function of both the input code and I_{OUTFS} and can be expressed as:

$$IOUTA = (DAC\ CODE/4096) \times I_{OUTFS} \quad (1)$$

$$IOUTB = (4095 - DAC\ CODE)/4096 \times I_{OUTFS} \quad (2)$$

where *DAC CODE* = 0 to 4095 (i.e., Decimal Representation).

As mentioned previously, I_{OUTFS} is a function of the reference current I_{REF}, which is nominally set by a reference voltage V_{REFIO} and external resistor R_{SET}. It can be expressed as:

$$I_{OUTFS} = 32 \times I_{REF} \quad (3)$$

where $I_{REF} = V_{REFIO}/R_{SET}$ (4)

The two current outputs will typically drive a resistive load directly or via a transformer. If dc coupling is required, IOUTA and IOUTB should be directly connected to matching resistive loads, R_{LOAD}, which are tied to analog common, ACOM. Note, R_{LOAD} may represent the equivalent load resistance seen by IOUTA or IOUTB as would be the case in a doubly terminated 50 Ω or 75 Ω cable. The single-ended voltage output appearing at the IOUTA and IOUTB nodes is simply :

$$V_{OUTA} = IOUTA \times R_{LOAD} \quad (5)$$

$$V_{OUTB} = IOUTB \times R_{LOAD} \quad (6)$$

Note the full-scale value of V_{OUTA} and V_{OUTB} should not exceed the specified output compliance range to maintain specified distortion and linearity performance.

The differential voltage, V_{DIFF}, appearing across IOUTA and IOUTB is:

$$V_{DIFF} = (IOUTA - IOUTB) \times R_{LOAD} \quad (7)$$

Substituting the values of I_{OUTA}, I_{OUTB}, and I_{REF}; V_{DIFF} can be expressed as:

$$V_{DIFF} = \{(2\ DAC\ CODE - 4095)/4096\} \times (32\ R_{LOAD}/R_{SET}) \times V_{REFIO} \quad (8)$$

These last two equations highlight some of the advantages of operating the AD9752 differentially. First, the differential operation will help cancel common-mode error sources associated with I_{OUTA} and I_{OUTB} such as noise, distortion and dc offsets. Second, the differential code dependent current and subsequent voltage, V_{DIFF}, is twice the value of the single-ended voltage output (i.e., V_{OUTA} or V_{OUTB}), thus providing twice the signal power to the load.

Note, the gain drift temperature performance for a single-ended (V_{OUTA} and V_{OUTB}) or differential output (V_{DIFF}) of the AD9752 can be enhanced by selecting temperature tracking resistors for R_{LOAD} and R_{SET} due to their ratiometric relationship as shown in Equation 8.

REFERENCE OPERATION

The AD9752 contains an internal 1.20 V bandgap reference that can easily be disabled and overridden by an external reference. REFIO serves as either an *input* or *output* depending on whether the internal or an external reference is selected. If REFLO is tied to ACOM, as shown in Figure 18, the internal reference is activated and REFIO provides a 1.20 V output. In this case, the internal reference *must* be compensated externally with a ceramic chip capacitor of 0.1 μF or greater from REFIO to REFLO. Also, REFIO should be buffered with an external amplifier having an input bias current less than 100 nA if any additional loading is required.

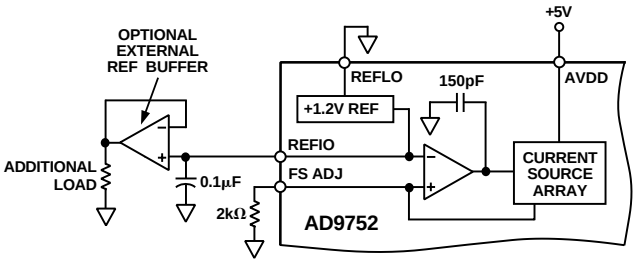


Figure 18. Internal Reference Configuration

The internal reference can be disabled by connecting REFLO to AVDD. In this case, an external reference may then be applied to REFIO as shown in Figure 19. The external reference may provide either a fixed reference voltage to enhance accuracy and drift performance or a varying reference voltage for gain control. Note that the 0.1 μF compensation capacitor is not required since the internal reference is disabled, and the high input impedance (i.e., 1 MΩ) of REFIO minimizes any loading of the external reference.

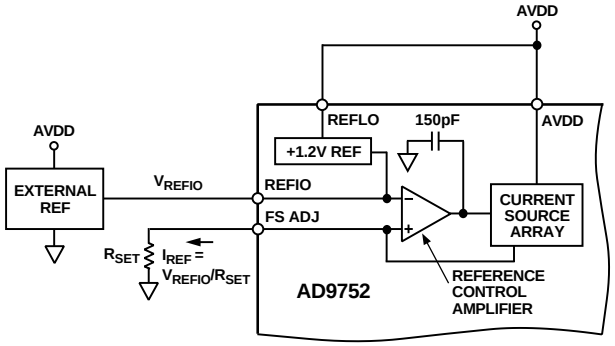


Figure 19. External Reference Configuration

REFERENCE CONTROL AMPLIFIER

The AD9752 also contains an internal control amplifier that is used to regulate the DAC's full-scale output current, I_{OUTFS}. The control amplifier is configured as a V-I converter as shown in Figure 19, such that its current output, I_{REF}, is determined by the ratio of the V_{REFIO} and an external resistor, R_{SET}, as stated in Equation 4. I_{REF} is copied over to the segmented current sources with the proper scaling factor to set I_{OUTFS} as stated in Equation 3.

The control amplifier allows a wide (10:1) adjustment span of I_{OUTFS} over a 2 mA to 20 mA range by setting IREF between 62.5 μA and 625 μA. The wide adjustment span of I_{OUTFS} provides several application benefits. The first benefit relates directly to the power dissipation of the AD9752, which is proportional to I_{OUTFS} (refer to the Power Dissipation section). The second benefit relates to the 20 dB adjustment, which is useful for system gain control purposes.

The small signal bandwidth of the reference control amplifier is approximately 0.5 MHz. The output of the control amplifier is internally compensated via a 150 pF capacitor that limits the control amplifier small-signal bandwidth and reduces its output impedance. Since the –3 dB bandwidth corresponds to the dominant pole, and hence the time constant, the settling time of the control amplifier to a stepped reference input response can be approximated. In this case, the time constant can be approximated to be 320 ns.

There are two methods in which I_{REF} can be varied for a fixed R_{SET}. The first method is suitable for a single-supply system in which the internal reference is disabled, and the common-mode voltage of REFIO is varied over its compliance range of 1.25 V to 0.10 V. REFIO can be driven by a single-supply amplifier or DAC, thus allowing I_{REF} to be varied for a fixed R_{SET}. Since the input impedance of REFIO is approximately 1 MΩ, a simple, low cost R-2R ladder DAC configured in the voltage mode topology may be used to control the gain. This circuit is shown in Figure 20 using the AD7524 and an external 1.2 V reference, the AD1580.

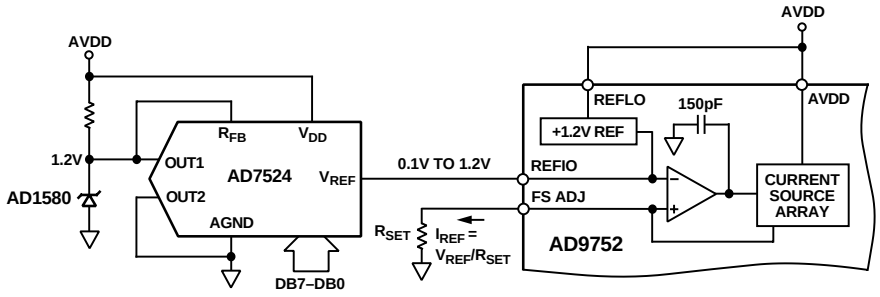


Figure 20. Single-Supply Gain Control Circuit

The second method may be used in a dual-supply system in which the common-mode voltage of REFIO is fixed and I_{REF} is varied by an external voltage, V_{GC}, applied to R_{SET} via an amplifier. An example of this method is shown in Figure 21, in which the internal reference is used to set the common-mode voltage of the control amplifier to 1.20 V. The external voltage, V_{GC}, is referenced to ACOM and should not exceed 1.2 V. The value of R_{SET} is such that I_{REFMAX} and I_{REFMIN} do not exceed 62.5 μA and 625 μA, respectively. The associated equations in Figure 21 can be used to determine the value of R_{SET}.

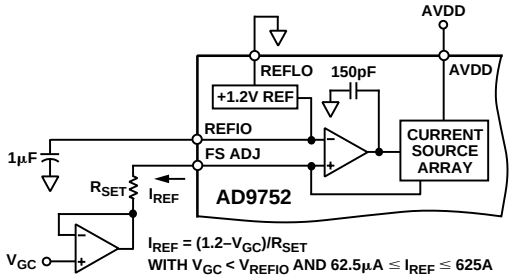


Figure 21. Dual-Supply Gain Control Circuit

ANALOG OUTPUTS

The AD9752 produces two complementary current outputs, IOUTA and IOUTB, which may be configured for single-end or differential operation. IOUTA and IOUTB can be converted into complementary single-ended voltage outputs, V_{OUTA} and V_{OUTB}, via a load resistor, R_{LOAD}, as described in the DAC Transfer Function section by Equations 5 through 8. The differential voltage, V_{DIFF}, existing between V_{OUTA} and V_{OUTB} can also be converted to a single-ended voltage via a transformer or differential amplifier configuration.

Figure 22 shows the equivalent analog output circuit of the AD9752 consisting of a parallel combination of PMOS differential current switches associated with each segmented current source. The output impedance of IOUTA and IOUTB is determined by the equivalent parallel combination of the PMOS switches and is typically 100 kΩ in parallel with 5 pF. Due to the nature of a PMOS device, the output impedance is also slightly dependent on the output voltage (i.e., V_{OUTA} and V_{OUTB}) and, to a lesser extent, the analog supply voltage, AVDD, and full-scale current, I_{OUTFS}. Although the output impedance's signal dependency can be a source of dc nonlinearity and ac linearity (i.e., distortion), its effects can be limited if certain precautions are noted.

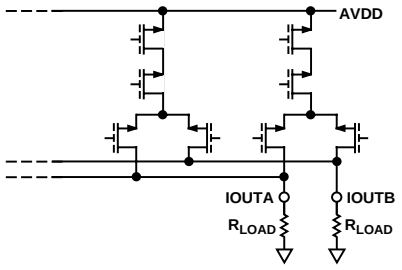


Figure 22. Equivalent Analog Output

IOUTA and IOUTB also have a negative and positive voltage compliance range. The negative output compliance range of –1.0 V is set by the breakdown limits of the CMOS process. Operation beyond this maximum limit may result in a breakdown of the output stage and affect the reliability of the AD9752. The positive output compliance range is slightly dependent on the full-scale output current, I_{OUTFS}. It degrades slightly from its nominal 1.25 V for an I_{OUTFS} = 20 mA to 1.00 V for an I_{OUTFS} = 2 mA. Operation beyond the positive compliance range will induce clipping of the output signal which severely degrades the AD9752's linearity and distortion performance.

For applications requiring the optimum dc linearity, IOUTA and/or IOUTB should be maintained at a virtual ground via an I-V op amp configuration. Maintaining IOUTA and/or IOUTB at a virtual ground keeps the output impedance of the AD9752 fixed, significantly reducing its effect on linearity. However, it does not necessarily lead to the optimum distortion performance due to limitations of the I-V op amp. Note that the INL/DNL specifications for the AD9752 are measured in this manner using IOUTA. In addition, these dc linearity specifications remain virtually unaffected over the specified power supply range of 4.5 V to 5.5 V.

Operating the AD9752 with reduced voltage output swings at IOUTA and IOUTB in a differential or single-ended output configuration reduces the signal dependency of its output impedance thus enhancing distortion performance. Although the voltage compliance range of IOUTA and IOUTB extends from –1.0 V to +1.25 V, optimum distortion performance is achieved when the maximum full-scale signal at IOUTA and IOUTB does not exceed approximately 0.5 V. A properly selected transformer with a grounded center-tap will allow the AD9752 to provide the required power and voltage levels to different loads while maintaining reduced voltage swings at IOUTA and IOUTB. DC-coupled applications requiring a differential or single-ended output configuration should size R_{LOAD} accordingly. Refer to Applying the AD9752 section for examples of various output configurations.

The most significant improvement in the AD9752's distortion and noise performance is realized using a differential output configuration. The common-mode error sources of both IOUTA and IOUTB can be substantially reduced by the common-mode rejection of a transformer or differential amplifier. These common-mode error sources include even-order distortion products and noise. The enhancement in distortion performance becomes more significant as the reconstructed waveform's frequency content increases and/or its amplitude decreases.

The distortion and noise performance of the AD9752 is also slightly dependent on the analog and digital supply as well as the full-scale current setting, I_{OUTFS}. Operating the analog supply at 5.0 V ensures maximum headroom for its internal PMOS current sources and differential switches leading to improved distortion performance. Although I_{OUTFS} can be set between 2 mA and 20 mA, selecting an I_{OUTFS} of 20 mA will provide the best distortion and noise performance also shown in Figure 8. The noise performance of the AD9752 is affected by the digital supply (DVDD), output frequency, and increases with increasing clock rate as shown in Figure 11. Operating the AD9752 with low voltage logic levels between 3 V and 3.3 V will slightly reduce the amount of on-chip digital noise.

In summary, the AD9752 achieves the optimum distortion and noise performance under the following conditions:

(1) Differential Operation.

(2) Positive voltage swing at IOUTA and IOUTB limited to +0.5 V.

(3) I_{OUTFS} set to 20 mA.

(4) Analog Supply (AVDD) set at 5.0 V.

(5) Digital Supply (DVDD) set at 3.0 V to 3.3 V with appropriate logic levels.

Note that the ac performance of the AD9752 is characterized under the above mentioned operating conditions.

DIGITAL INPUTS

The AD9752's digital input consists of 12 data input pins and a clock input pin. The 12-bit parallel data inputs follow standard positive binary coding where DB11 is the most significant bit (MSB) and DB0 is the least significant bit (LSB). IOUTA produces a full-scale output current when all data bits are at Logic 1. IOUTB produces a complementary output with the full-scale current split between the two outputs as a function of the input code.

The digital interface is implemented using an edge-triggered master slave latch. The DAC output is updated following the rising edge of the clock as shown in Figure 1 and is designed to support a clock rate as high as 125 MSPS. The clock can be operated at any duty cycle that meets the specified latch pulsewidth. The setup and hold times can also be varied within the clock cycle as long as the specified minimum times are met; although the location of these transition edges may affect digital feedthrough and distortion performance. *Best performance is typically achieved when the input data transitions on the falling edge of a 50% duty cycle clock.*

The digital inputs are CMOS compatible with logic thresholds, $V_{THRESHOLD}$ set to approximately half the digital positive supply (DVDD) or

$$V_{THRESHOLD} = DVDD/2\ (\pm 20\%)$$

The internal digital circuitry of the AD9752 is capable of operating over a digital supply range of 2.7 V to 5.5 V. As a result, the digital inputs can also accommodate TTL levels when DVDD is set to accommodate the maximum high level voltage of the TTL drivers $V_{OH(MAX)}$. A DVDD of 3 V to 3.3 V will typically ensure proper compatibility with most TTL logic families. Figure 23 shows the equivalent digital input circuit for the data and clock inputs. The sleep mode input is similar with the exception that it contains an active pull-down circuit, thus ensuring that the AD9752 remains enabled if this input is left disconnected.

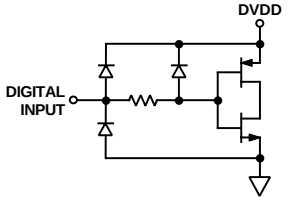


Figure 23. Equivalent Digital Input

Since the AD9752 is capable of being updated up to 125 MSPS, the quality of the clock and data input signals are important in achieving the optimum performance. The drivers of the digital data interface circuitry should be specified to meet the minimum setup and hold times of the AD9752 as well as its required min/max input logic level thresholds. Typically, the selection of the slowest logic family that satisfies the above conditions will result in the lowest data feedthrough and noise.

Digital signal paths should be kept short and run lengths matched to avoid propagation delay mismatch. The insertion of a low value resistor network (i.e., 20 Ω to 100 Ω) between the AD9752 digital inputs and driver outputs may be helpful in reducing any overshooting and ringing at the digital inputs that contribute to data feedthrough. For longer run lengths and high data update rates, strip line techniques with proper termination resistors should be considered to maintain "clean" digital inputs. Also, operating the AD9752 with reduced logic swings and a corresponding digital supply (DVDD) will also reduce data feedthrough.

The external clock driver circuitry should provide the AD9752 with a low jitter clock input meeting the min/max logic levels while providing fast edges. Fast clock edges will help minimize any jitter that will manifest itself as phase noise on a reconstructed waveform. Thus, the clock input should be driven by the fastest logic family suitable for the application.

Note, the clock input could also be driven via a sine wave, which is centered around the digital threshold (i.e., DVDD/2), and meets the min/max logic threshold. This will typically result in a slight degradation in the phase noise, which becomes more noticeable at higher sampling rates and output frequencies. Also, at higher sampling rates, the 20% tolerance of the digital logic threshold should be considered since it will affect the effective clock duty cycle and subsequently cut into the required data setup and hold times.

INPUT CLOCK/DATA TIMING RELATIONSHIP

SNR in a DAC is dependent on the relationship between the position of the clock edges and the point in time at which the input data changes. The AD9752 is positive edge triggered, and so exhibits SNR sensitivity when the data transition is close to this edge. In general, the goal when applying the AD9752 is to make the data transitions shortly after the positive clock edge. This becomes more important as the sample rate increases. Figure 24 shows the relationship of SNR to clock placement with different sample rates and different frequencies out. Note that at the lower sample rates, much more tolerance is allowed in clock placement, while at higher rates, much more care must be taken.

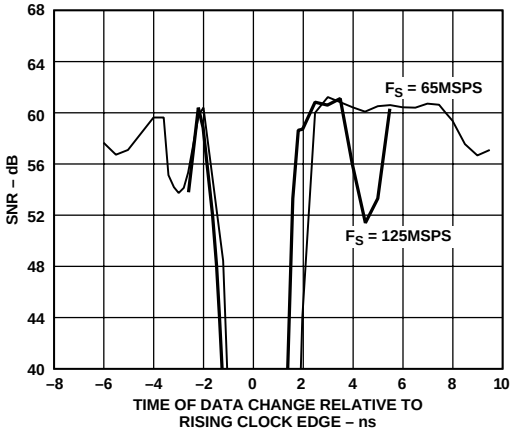


Figure 24. SNR vs. Clock Placement @ f_{OUT} = 10 MHz

SLEEP MODE OPERATION

The AD9752 has a power-down function which turns off the output current and reduces the supply current to less than 8.5 mA over the specified supply range of 2.7 V to 5.5 V and temperature range. This mode can be activated by applying a logic level "1" to the SLEEP pin. This digital input also contains an active pull-down circuit that ensures the AD9752 remains enabled if this input is left disconnected. The AD9752 takes less than 50 ns to power down and approximately 5 µs to power back up.

POWER DISSIPATION

The power dissipation, P_D, of the AD9752 is dependent on several factors which include: (1) AVDD and DVDD, the power supply voltages; (2) I_{OUTFS}, the full-scale current output; (3) f_{CLOCK}, the update rate; (4) and the reconstructed digital input waveform. The power dissipation is directly proportional to the analog supply current, I_{AVDD}, and the digital supply current, I_{DVDD}. I_{AVDD} is directly proportional to I_{OUTFS} as shown in Figure 25 and is insensitive to f_{CLOCK}.

Conversely, I_{DVDD} is dependent on both the digital input waveform, f_{CLOCK}, and digital supply DVDD. Figures 26 and 27 show I_{DVDD} as a function of full-scale sine wave output ratios (f_{OUT}/f_{CLOCK}) for various update rates with DVDD = 5 V and DVDD = 3 V, respectively. Note, how I_{DVDD} is reduced by more than a factor of 2 when DVDD is reduced from 5 V to 3 V.

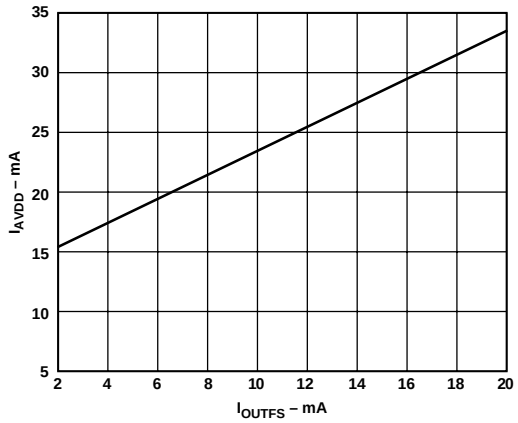


Figure 25. I_{AVDD} vs. I_{OUTFS}

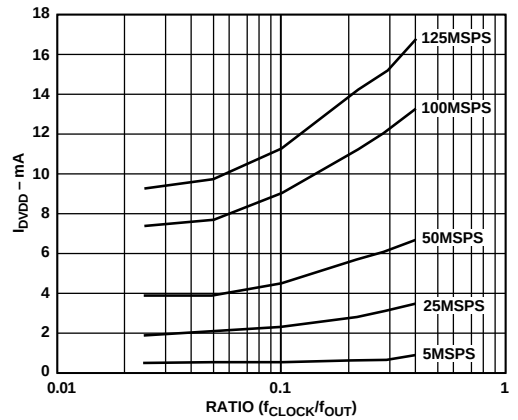


Figure 26. I_{DVDD} vs. Ratio @ DVDD = 5 V

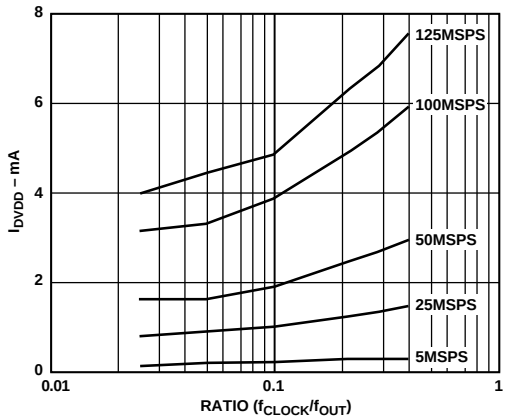


Figure 27. I_{DVDD} vs. Ratio @ DVDD = 3 V

APPLYING THE AD9752 OUTPUT CONFIGURATIONS

The following sections illustrate some typical output configurations for the AD9752. Unless otherwise noted, it is assumed that I_{OUTFS} is set to a nominal 20 mA. For applications requiring the optimum dynamic performance, a differential output configuration is suggested. A differential output configuration may consist of either an RF transformer or a differential op amp configuration. The transformer configuration provides the optimum high frequency performance and is recommended for any application allowing for ac coupling. The differential op amp configuration is suitable for applications requiring dc coupling, a bipolar output, signal gain and/or level shifting.

A single-ended output is suitable for applications requiring a unipolar voltage output. A positive unipolar output voltage will result if IOUTA and/or IOUTB is connected to an appropriately sized load resistor, R_{LOAD}, referred to ACOM. This configuration may be more suitable for a single-supply system requiring a dc coupled, ground referred output voltage. Alternatively, an amplifier could be configured as an I-V converter thus converting IOUTA or IOUTB into a negative unipolar voltage. This configuration provides the best dc linearity since IOUTA or IOUTB is maintained at a virtual ground. Note, IOUTA provides slightly better performance than IOUTB.

DIFFERENTIAL COUPLING USING A TRANSFORMER

An RF transformer can be used to perform a differential-to-single-ended signal conversion as shown in Figure 28. A differentially coupled transformer output provides the optimum distortion performance for output signals whose spectral content lies within the transformer's passband. An RF transformer such as the Mini-Circuits T1-1T provides excellent rejection of common-mode distortion (i.e., even-order harmonics) and noise over a wide frequency range. It also provides electrical isolation and the ability to deliver twice the power to the load. Transformers with different impedance ratios may also be used for impedance matching purposes. Note that the transformer provides ac coupling only.

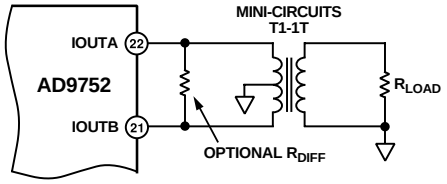


Figure 28. Differential Output Using a Transformer

The center tap on the primary side of the transformer must be connected to ACOM to provide the necessary dc current path for both IOUTA and IOUTB. The complementary voltages appearing at IOUTA and IOUTB (i.e., V_{OUTA} and V_{OUTB}) swing symmetrically around ACOM and should be maintained with the specified output compliance range of the AD9752. A differential resistor, R_{DIFF}, may be inserted in applications in which the output of the transformer is connected to the load, R_{LOAD}, via a passive reconstruction filter or cable. R_{DIFF} is determined by the transformer's impedance ratio and provides the proper source termination which results in a low VSWR. Note that approximately half the signal power will be dissipated across R_{DIFF}.

DIFFERENTIAL USING AN OP AMP

An op amp can also be used to perform a differential to single-ended conversion as shown in Figure 29. The AD9752 is configured with two equal load resistors, R_{LOAD}, of 25 Ω. The differential voltage developed across IOUTA and IOUTB is converted to a single-ended signal via the differential op amp configuration. An optional capacitor can be installed across IOUTA and IOUTB forming a real pole in a low-pass filter. The addition of this capacitor also enhances the op amps distortion performance by preventing the DACs high slewing output from overloading the op amp's input.

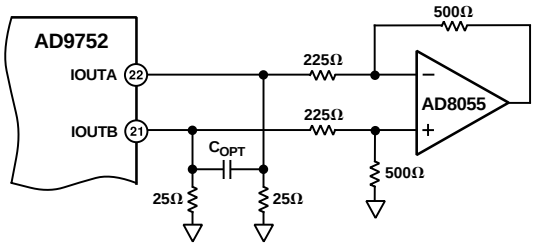


Figure 29. DC Differential Coupling Using an Op Amp

The common-mode rejection of this configuration is typically determined by the resistor matching. In this circuit, the differential op amp circuit is configured to provide some additional signal gain. The op amp must operate off of a dual supply since its output is approximately ±1.0 V. A high speed amplifier such as the AD8055 or AD9632 capable of preserving the differential performance of the AD9752 while meeting other system level objectives (i.e., cost, power) should be selected. The op amps differential gain, its gain setting resistor values, and full-scale output swing capabilities should all be considered when optimizing this circuit.

The differential circuit shown in Figure 30 provides the necessary level-shifting required in a single supply system. In this case, AVDD which is the positive analog supply for both the AD9752 and the op amp is also used to level-shift the differential output of the AD9752 to midsupply (i.e., AVDD/2). The AD8041 is a suitable op amp for this application.

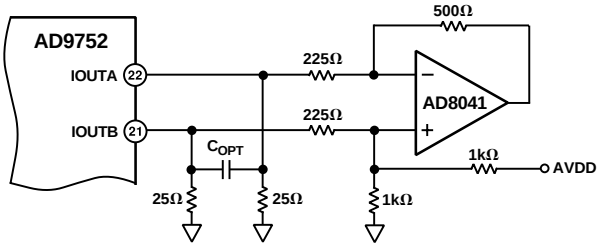


Figure 30. Single-Supply DC Differential Coupled Circuit

SINGLE-ENDED UNBUFFERED VOLTAGE OUTPUT

Figure 31 shows the AD9752 configured to provide a unipolar output range of approximately 0 V to +0.5 V for a doubly terminated 50 Ω cable since the nominal full-scale current, I_{OUTFS}, of 20 mA flows through the equivalent R_{LOAD} of 25 Ω. In this case, R_{LOAD} represents the equivalent load resistance seen by IOUTA or IOUTB. The unused output (IOUTA or IOUTB) can be connected to ACOM directly or via a matching R_{LOAD}. Different values of I_{OUTFS} and R_{LOAD} can be selected as long as the positive compliance range is adhered to. One additional consideration in this mode is the integral nonlinearity (INL) as discussed in the ANALOG OUTPUT section of this data sheet. For optimum INL performance, the single-ended, buffered voltage output configuration is suggested.

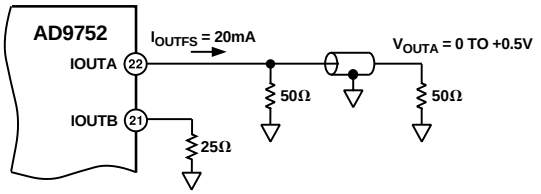


Figure 31. 0 V to +0.5 V Unbuffered Voltage Output

SINGLE-ENDED, BUFFERED VOLTAGE OUTPUT CONFIGURATION

Figure 32 shows a buffered single-ended output configuration in which the op amp U1 performs an I-V conversion on the AD9752 output current. U1 maintains IOUTA (or IOUTB) at a virtual ground, thus minimizing the nonlinear output impedance effect on the DAC's INL performance as discussed in the ANALOG OUTPUT section. Although this single-ended configuration typically provides the best dc linearity performance, its ac distortion performance at higher DAC update rates may be limited by U1's slewing capabilities. U1 provides a negative unipolar output voltage and its full-scale output voltage is simply the product of R_{FB} and I_{OUTFS}. The full-scale output should be set within U1's voltage output swing capabilities by scaling I_{OUTFS} and/or R_{FB}. An improvement in ac distortion performance may result with a reduced I_{OUTFS} since the signal current U1 will be required to sink will be subsequently reduced.

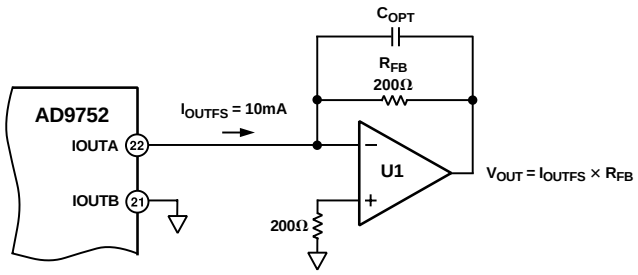


Figure 32. Unipolar Buffered Voltage Output

POWER AND GROUNDING CONSIDERATIONS, POWER SUPPLY REJECTION

Many applications seek high speed and high performance under less than ideal operating conditions. In these circuits, the implementation and construction of the printed circuit board design is as important as the circuit design. Proper RF techniques must be used for device selection, placement and routing as well as power supply bypassing and grounding to ensure optimum performance. Figures 42-47 illustrate the recommended printed circuit board ground, power and signal plane layouts which are implemented on the AD9752 evaluation board.

One factor that can measurably affect system performance is the ability of the DAC output to reject dc variations or ac noise superimposed on the analog or digital dc power distribution (i.e., AVDD, DVDD). This is referred to as Power Supply Rejection Ratio (PSRR). For dc variations of the power supply, the resulting performance of the DAC directly corresponds to a gain error associated with the DAC's full-scale current, I_{OUTFS}. AC noise on the dc supplies is common in applications where the power distribution is generated by a switching power supply. Typically, switching power supply noise will occur over the spectrum from tens of kHz to several MHz. PSRR vs. frequency of the AD9752 AVDD supply, over this frequency range, is given in Figure 33.

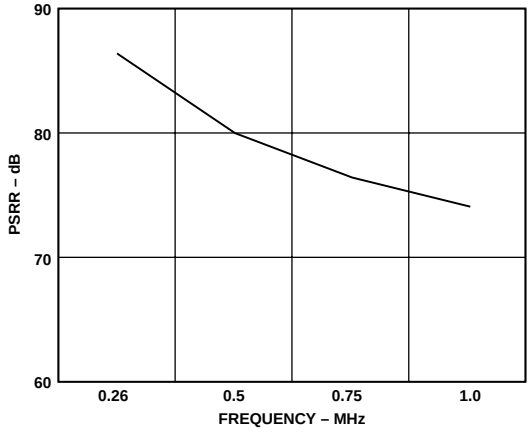


Figure 33. Power Supply Rejection Ratio of AD9752

Note that the units in Figure 33 are given in units of (amps out)/(volts in). Noise on the analog power supply has the effect of modulating the internal switches, and therefore the output current. The voltage noise on the dc power, therefore, will be added in a nonlinear manner to the desired I_{OUT}. Due to the relative different sizes of these switches, PSRR is very code dependent. This can produce a mixing effect which can modulate low frequency power supply noise to higher frequencies. Worst case PSRR for either one of the differential DAC outputs will occur when the full-scale current is directed towards that output. As a result, the PSRR measurement in Figure 33 represents a worst case condition in which the digital inputs remain static and the full scale output current of 20 mA is directed to the DAC output being measured.

An example serves to illustrate the effect of supply noise on the analog supply. Suppose a switching regulator with a switching frequency of 250 kHz produces 10 mV rms of noise and for simplicity sake (i.e., ignore harmonics), all of this noise is concentrated at 250 kHz. To calculate how much of this undesired noise will appear as current noise super imposed on the DAC's full-scale current, I_{OUTFS}, one must determine the PSRR in dB using Figure 33 at 250 kHz. To calculate the PSRR for a given R_{LOAD}, such that the units of PSRR are converted from A/V to V/V, adjust the curve in Figure 33 by the scaling factor $20 \times \text{Log}(R_{LOAD})$. For instance, if R_{LOAD} is 50 Ω, the PSRR is reduced by 34 dB (i.e., PSRR of the DAC at 1 MHz which is 74 dB in Figure 33 becomes 40 dB V_{OUT}/V_{IN}).

Proper grounding and decoupling should be a primary objective in any high speed, high resolution system. The AD9752 features separate analog and digital supply and ground pins to optimize the management of analog and digital ground currents in a system. In general, AVDD, the analog supply, should be decoupled to ACOM, the analog common, as close to the chip as physically possible. Similarly, DVDD, the digital supply, should be decoupled to DCOM as close as physically as possible.

For those applications that require a single +5 V or +3 V supply for both the analog and digital supply, a clean analog supply may be generated using the circuit shown in Figure 34. The circuit consists of a differential LC filter with separate power supply and return lines. Lower noise can be attained using low ESR type electrolytic and tantalum capacitors.

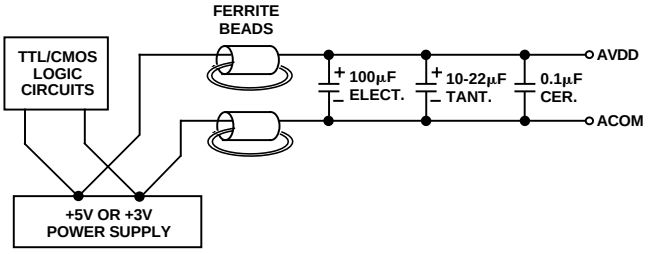


Figure 34. Differential LC Filter for Single +5 V or +3 V Applications

Maintaining low noise on power supplies and ground is critical to obtaining optimum results from the AD9752. If properly implemented, ground planes can perform a host of functions on high speed circuit boards: bypassing, shielding, current transport, etc. In mixed signal design, the analog and digital portions of the board should be distinct from each other, with the analog ground plane confined to the areas covering the analog signal traces, and the digital ground plane confined to areas covering the digital interconnects.

All analog ground pins of the DAC, reference and other analog components should be tied directly to the analog ground plane. The two ground planes should be connected by a path 1/8 to 1/4 inch wide underneath or within 1/2 inch of the DAC to

maintain optimum performance. Care should be taken to ensure that the ground plane is uninterrupted over crucial signal paths. On the digital side, this includes the digital input lines running to the DAC as well as any clock signals. On the analog side, this includes the DAC output signal, reference signal and the supply feeders.

The use of wide runs or planes in the routing of power lines is also recommended. This serves the dual role of providing a low series impedance power supply to the part, as well as providing some "free" capacitive decoupling to the appropriate ground plane. It is essential that care be taken in the layout of signal and power ground interconnects to avoid inducing extraneous voltage drops in the signal ground paths. It is recommended that all connections be short, direct and as physically close to the package as possible in order to minimize the sharing of conduction paths between different currents. When runs exceed an inch in length, strip line techniques with proper termination resistor should be considered. The necessity and value of this resistor will be dependent upon the logic family used.

For a more detailed discussion of the implementation and construction of high speed, mixed signal printed circuit boards, refer to Analog Devices' application notes AN-280 and AN-333.

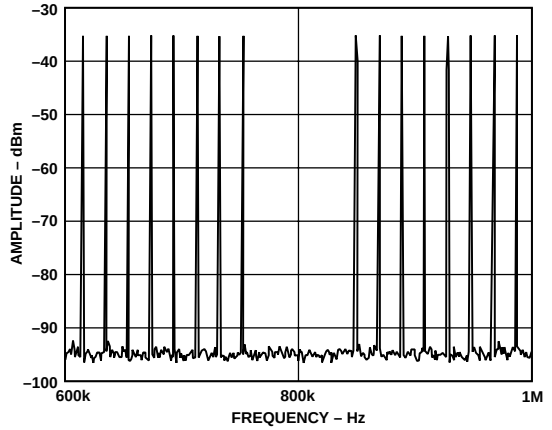


Figure 35a. Notch in Missing Bin at 750 kHz is Down >60 dB. (Peak Amplitude + 0 dBm).

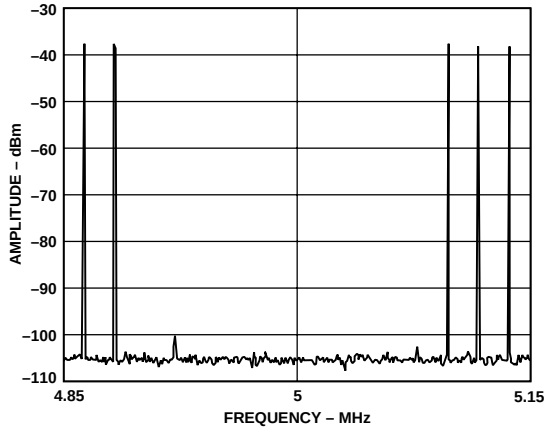


Figure 35b. Notch in Missing Bin at 5 MHz is Down >60 dB. (Peak Amplitude + 0 dBm).

APPLICATIONS

VDSL Applications Using the AD9752

Very High Frequency Digital Subscriber Line (VDSL) technology is growing rapidly in applications requiring data transfer over relatively short distances. By using QAM modulation and transmitting the data in multiple discrete tones, high data rates can be achieved.

As with other multitone applications, each VDSL tone is capable of transmitting a given number of bits, depending on the signal-to-noise ratio (SNR) in a narrow band around that tone. The tones are evenly spaced over the range of several kHz to 10 MHz. At the high frequency end of this range, performance is generally limited by cable characteristics and environmental factors, such as external interferers. Performance at the lower frequencies is much more dependent on the performance of the components in the signal chain. In addition to in-band noise, intermodulation from other tones can also potentially interfere with the recovery of data for a given tone. The two graphs in Figure 35 represent a 500 tone missing bin test vector, with frequencies evenly spaced from 400 Hz to 10 MHz. This test is very commonly done to determine if distortion will limit the number of bits which can be transmitted in a tone. The test vector has a series of missing tones around 750 kHz, which is represented in Figure 35a and a series of missing tones around 5 MHz which is represented in Figure 35b. In both cases, the spurious free range between the transmitted tones and the empty bins is greater than 60 dB.

Using the AD9752 for Quadrature Amplitude Modulation (QAM)

QAM is one of the most widely used digital modulation schemes in digital communication systems. This modulation technique can be found in FDM as well as spreadspectrum (i.e., CDMA) based systems. A QAM signal is a carrier frequency that is modulated in both amplitude (i.e., AM modulation) and phase (i.e., PM modulation). It can be generated by independently modulating two carriers of identical frequency but with a 90° phase difference. This results in an in-phase (I) carrier component and a quadrature (Q) carrier component at a 90° phase shift with respect to the I component. The I and Q components are then summed to provide a QAM signal at the specified carrier frequency.

A common and traditional implementation of a QAM modulator is shown in Figure 36. The modulation is performed in the analog domain in which two DACs are used to generate the baseband I and Q components, respectively. Each component is then typically applied to a Nyquist filter before being applied to a quadrature mixer. The matching Nyquist filters shape and limit each component's spectral envelope while minimizing intersymbol interference. The DAC is typically updated at the QAM symbol rate or possibly a multiple of it if an interpolating filter precedes the DAC. The use of an interpolating filter typically eases the implementation and complexity of the analog filter, which can be a significant contributor to mismatches in gain and phase between the two baseband channels. A quadrature mixer modulates the I and Q components with in-phase and quadrature phase carrier frequency and then sums the two outputs to provide the QAM signal.

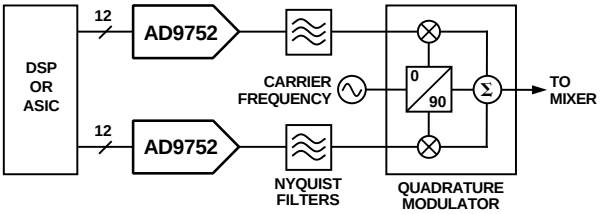


Figure 36. Typical Analog QAM Architecture

In this implementation, it is much more difficult to maintain proper gain and phase matching between the I and Q channels. The circuit implementation shown in Figure 37 helps improve upon the matching and temperature stability characteristics between the I and Q channels, as well as showing a path for upconversion using the AD8346 quadrature modulator. Using a single voltage reference derived from U1 to set the gain for both the I and Q channels will improve the gain matching and stability. R_{CAL} can be used to compensate for any mismatch in gain between the two channels. This mismatch may be attributed to the mismatch between R_{SET1} and R_{SET2}, effective load resistance of each channel, and/or the voltage offset of the control amplifier in each DAC. The differential voltage outputs of U1 and U2 are fed into the respective differential inputs of the AD8346 via matching networks.

Using the same matching techniques described above, Figure 38 shows an example of the AD9752 used in a W-CDMA transmitter application using the AD6122 CDMA 3 V transmitter IF subsystem. The AD6122 has functions, such as external gain control and low distortion characteristics, needed for the superior Adjacent Channel Power (ACP) requirements of W-CDMA.

CDMA

Carrier Division Multiple Access, or CDMA, is an air transmit/receive scheme where the signal in the transmit path is modulated with a pseudorandom digital code (sometimes referred to as the spreading code). The effect of this is to spread the transmitted signal across a wide spectrum. Similar to a DMT waveform, a CDMA waveform containing multiple subscribers can be characterized as having a high peak to average ratio (i.e., crest factor), thus demanding highly linear components in the transmit signal path. The bandwidth of the spectrum is defined by the CDMA standard being used, and in operation is implemented by using a spreading code with particular characteristics.

Distortion in the transmit path can lead to power being transmitted out of the defined band. The ratio of power transmitted in-band to out-of-band is often referred to as Adjacent Channel Power (ACP). This is a regulatory issue due to the possibility of interference with other signals being transmitted by air. Regulatory bodies define a spectral mask outside of the transmit band, and the ACP must fall under this mask. If distortion in the transmit path cause the ACP to be above the spectral mask, then filtering, or different component selection is needed to meet the mask requirements.

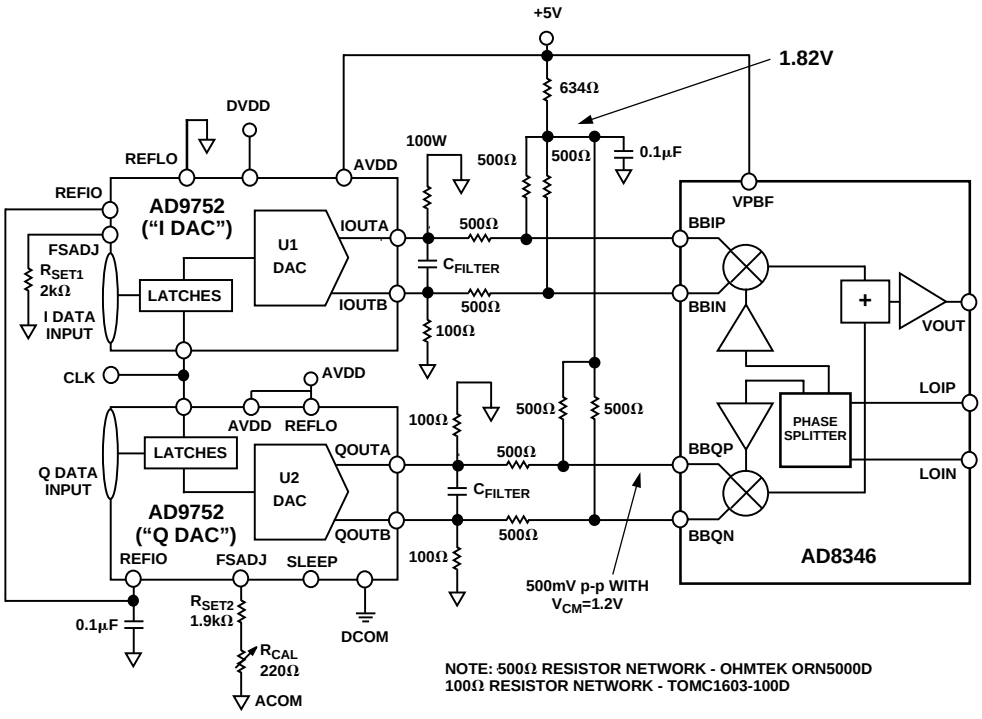


Figure 37. Baseband QAM Implementation Using Two AD9752s

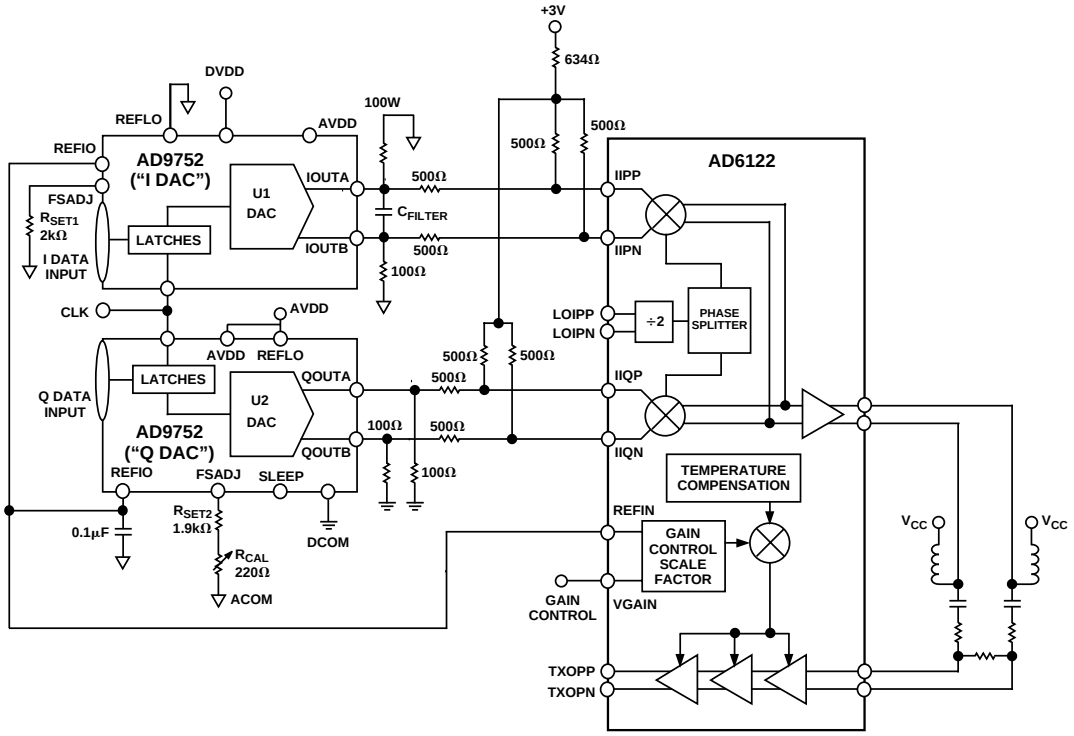


Figure 38. CDMA Transmit Application Using AD9752

Figure 39 shows the AD9752 reconstructing a wideband, or W-CDMA test vector with a bandwidth of 5 MHz, centered at 15.625 MHz and being sampled at 62.5 MSPS. ACP for the given test vector is measured at 70 dB.

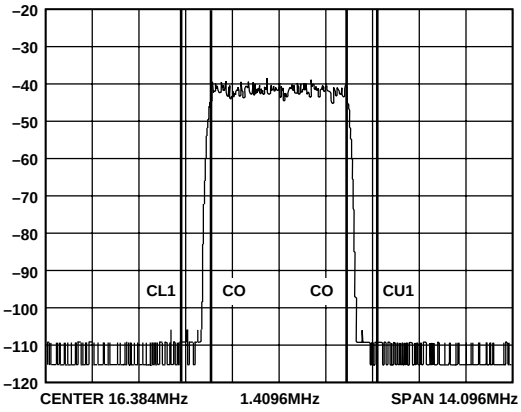


Figure 39. CDMA Signal, Sampled at 65 MSPS, Adjacent Channel Power >70 dBm

It is also possible to generate a QAM signal completely in the digital domain via a DSP or ASIC, in which case only a single DAC of sufficient resolution and performance is required to reconstruct the QAM signal. Also available from several vendors are Digital ASICs which implement other digital modulation schemes such as PSK and FSK. This digital implementation has the benefit of generating perfectly matched I and Q components in terms of gain and phase, which is essential in maintaining optimum performance in a communication system. In this implementation, the reconstruction DAC must be operating at a sufficiently high clock rate to accommodate the highest specified QAM carrier frequency. Figure 40 shows a block diagram of such an implementation using the AD9752.

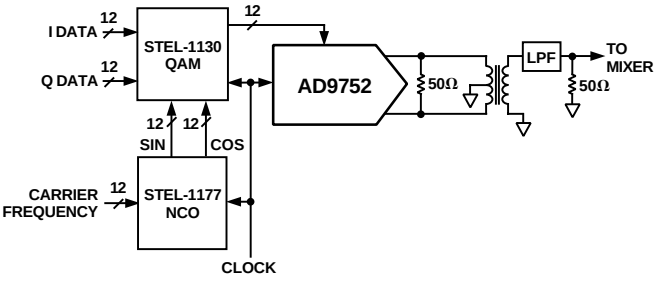


Figure 40. Digital QAM Architecture

AD9752 EVALUATION BOARD

General Description

The AD9752-EB is an evaluation board for the AD9752 12-bit D/A converter. Careful attention to layout and circuit design combined with a prototyping area allow the user to easily and effectively evaluate the AD9752 in any application where high resolution, high speed conversion is required.

This board allows the user the flexibility to operate the AD9752 in various configurations. Possible output configurations include transformer coupled, resistor terminated, inverting/noninverting and differential amplifier outputs. The digital inputs are designed to be driven directly from various word generators, with the on-board option to add a resistor network for proper load termination. Provisions are also made to operate the AD9752 with either the internal or external reference, or to exercise the power-down feature.

Refer to the application note AN-420 for a thorough description and operating instructions for the AD9752 evaluation board.

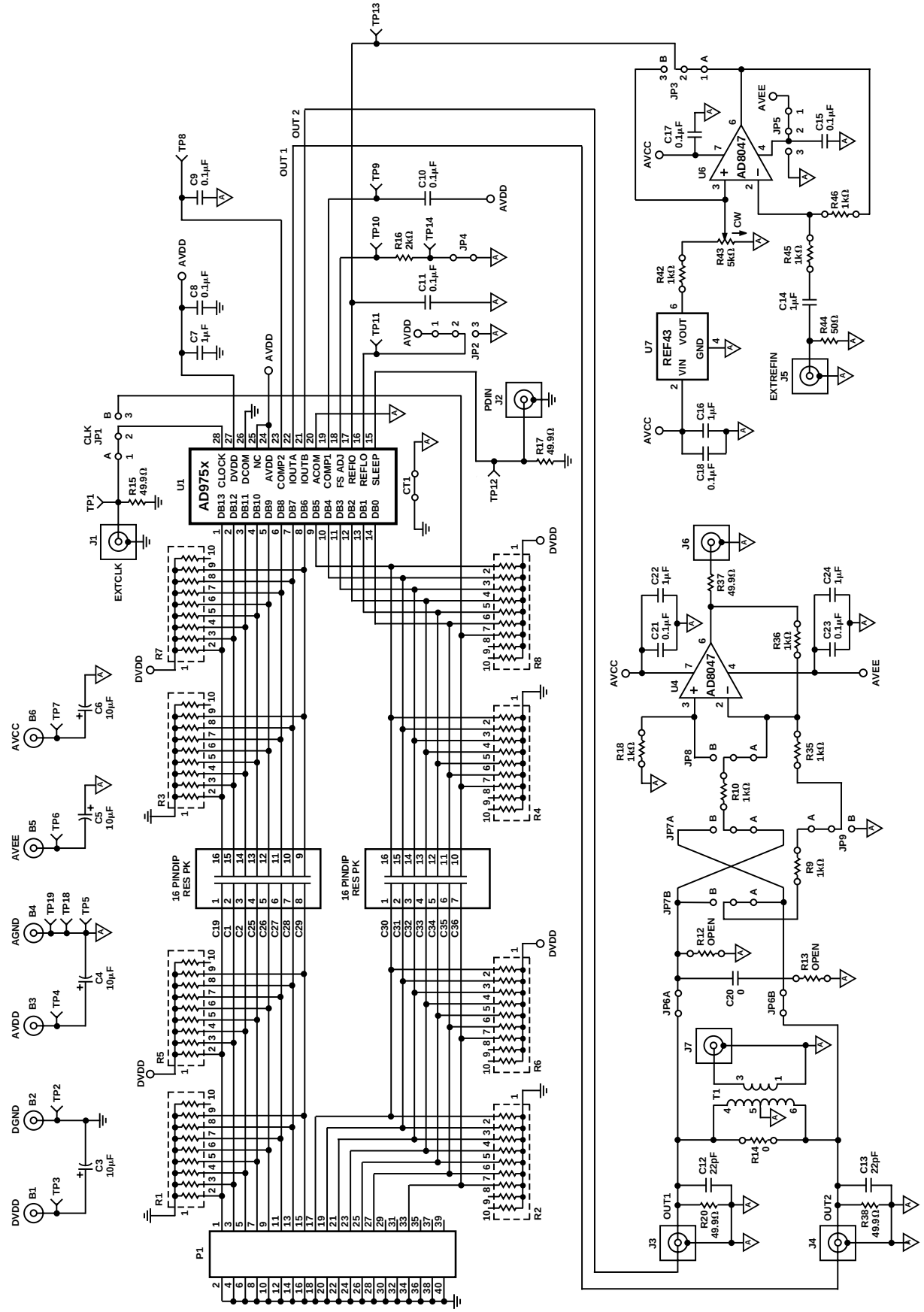


Figure 41. Evaluation Board Schematic

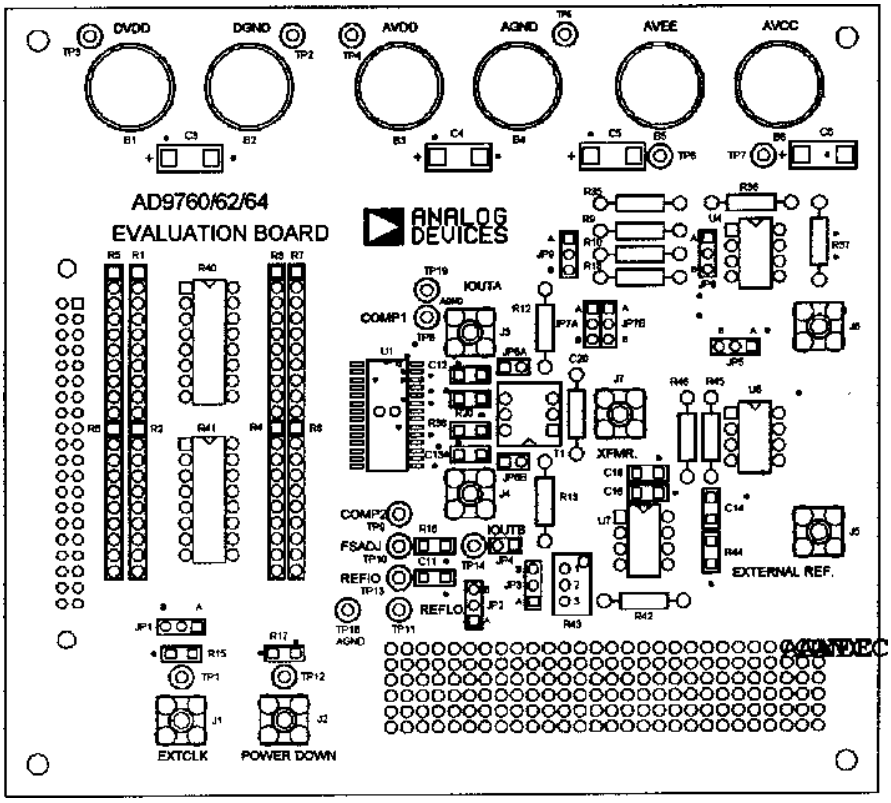


Figure 42. Silkscreen Layer—Top

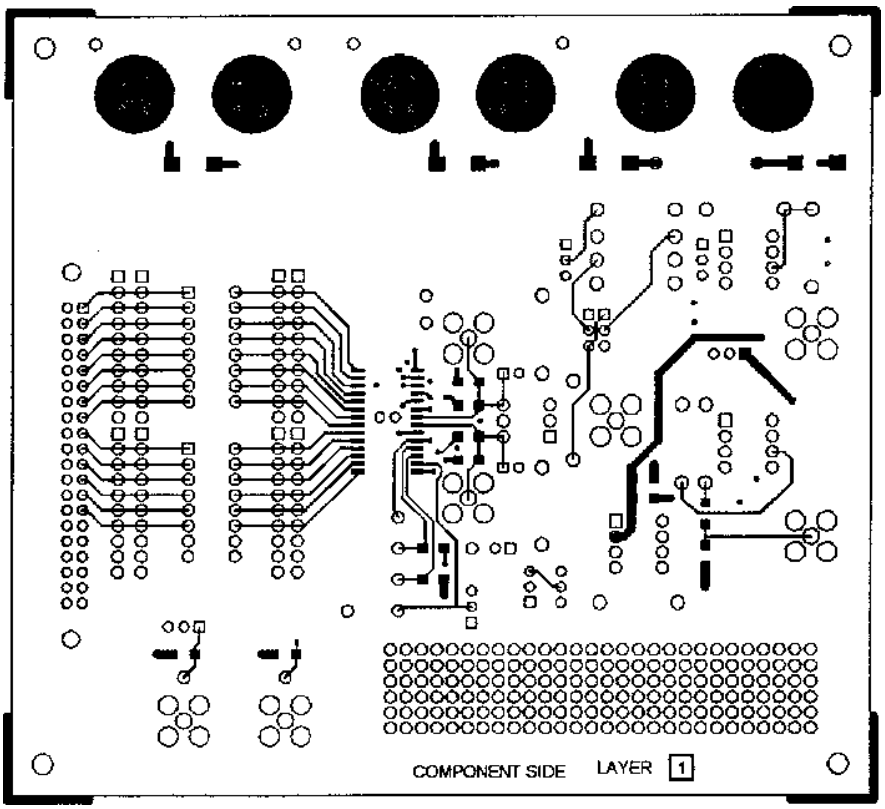


Figure 43. Component Side PCB Layout (Layer 1)

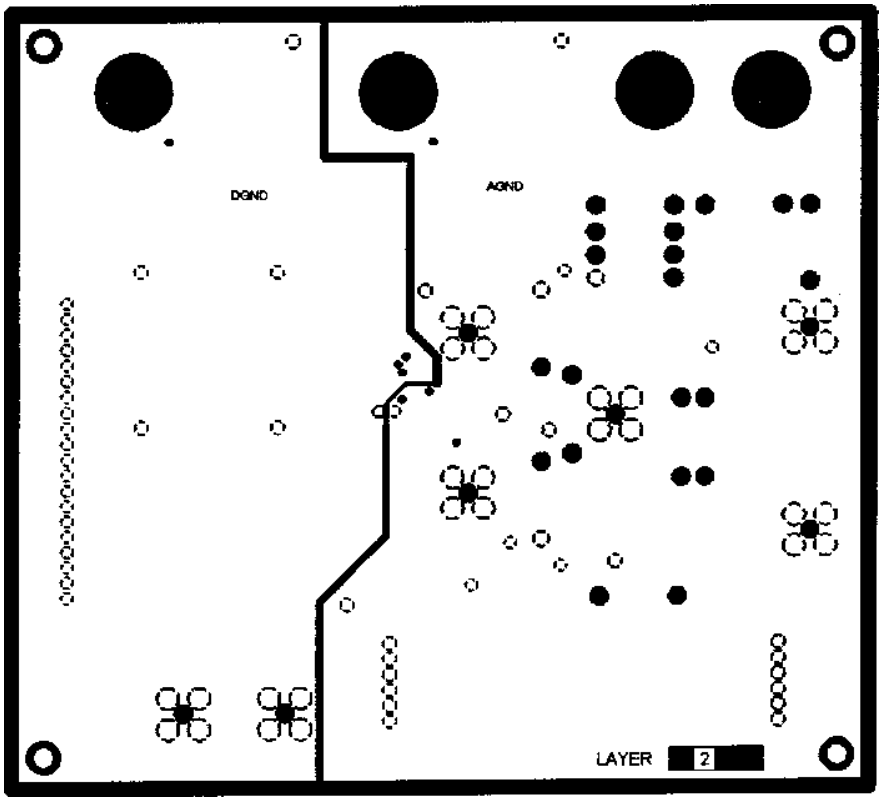


Figure 44. Ground Plane PCB Layout (Layer 2)

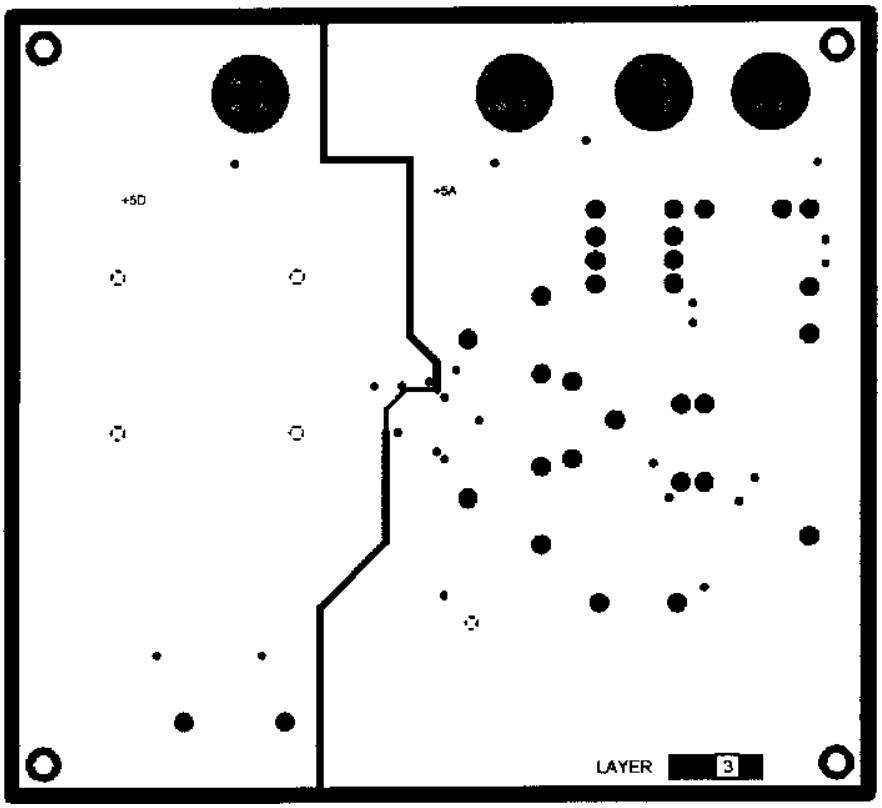


Figure 45. Power Plane PCB Layout (Layer 3)

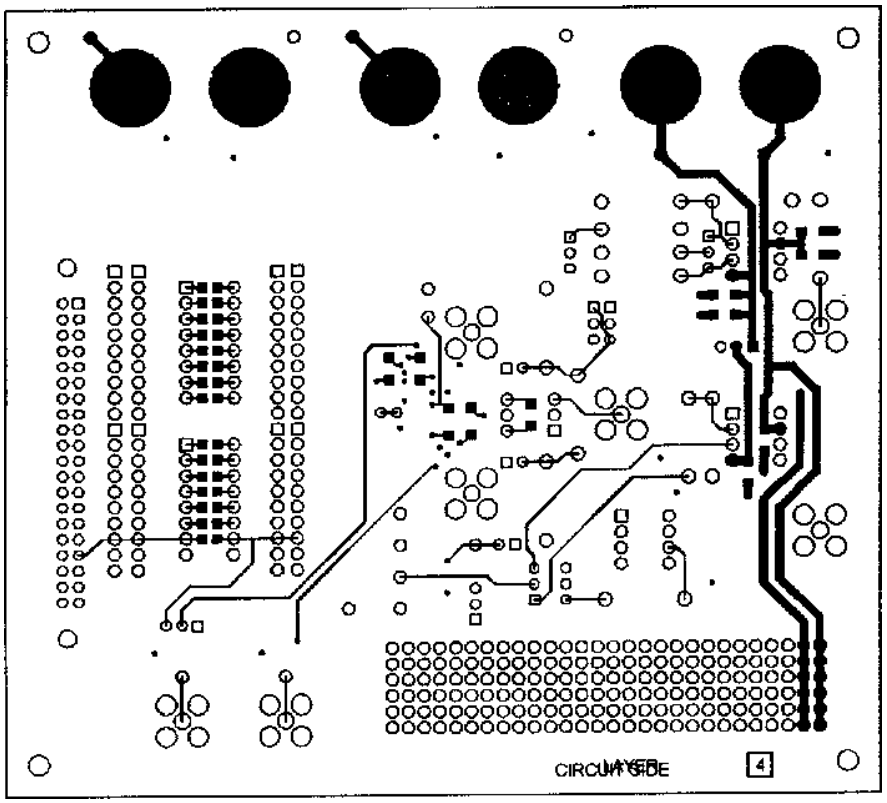


Figure 46. Solder Side PCB Layout (Layer 4)

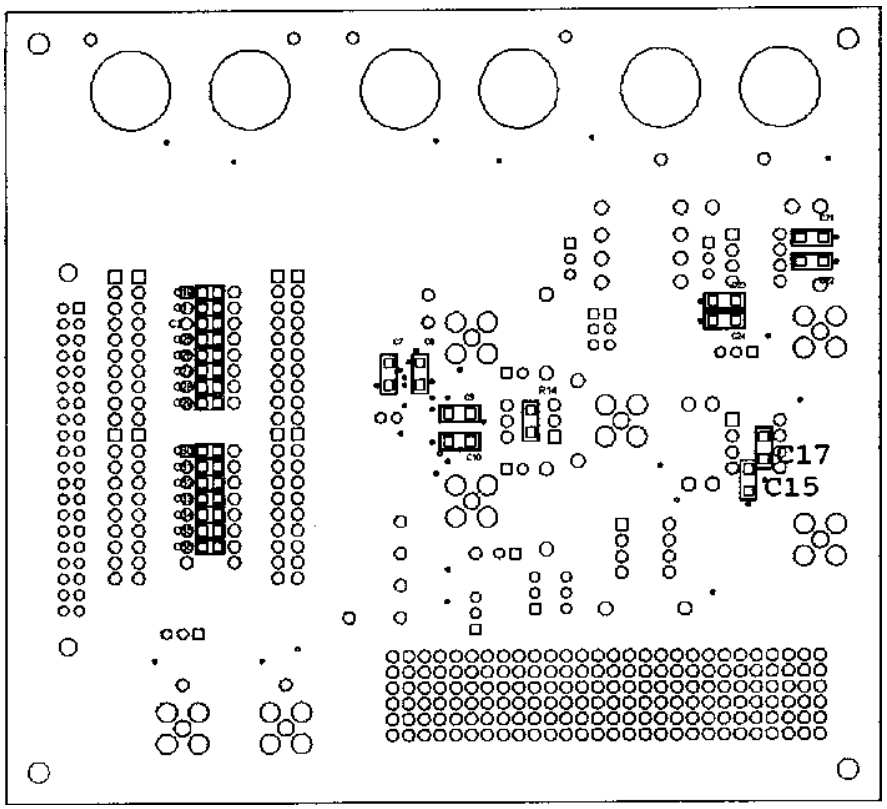
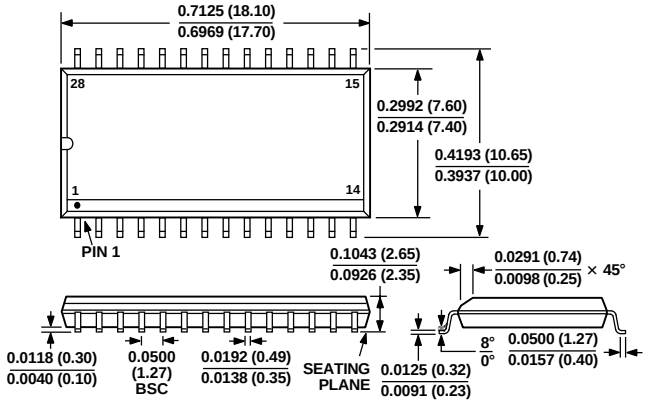


Figure 47. Silkscreen Layer—Bottom

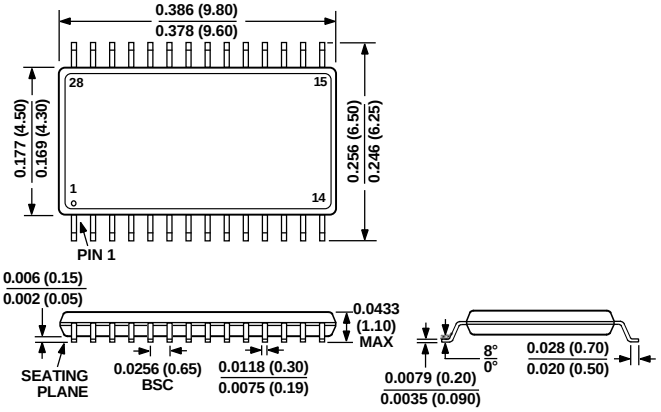
OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).

28-Lead, 300 Mil SOIC
(R-28)



28-Lead TSSOP
(RU-28)



C3332–8–1/99

PRINTED IN U.S.A.

ADS805

SBAS073B – JANUARY 1997 – REVISED NOVEMBER 2002

12-Bit, 20MHz Sampling ANALOG-TO-DIGITAL CONVERTER

FEATURES

- **HIGH SFDR: 74dB at 9.8MHz f_{IN}**
- **HIGH SNR: 68dB**
- **LOW POWER: 300mW**
- **LOW DLE: 0.25LSB**
- **FLEXIBLE INPUT RANGE**
- **OVER-RANGE INDICATOR**

APPLICATIONS

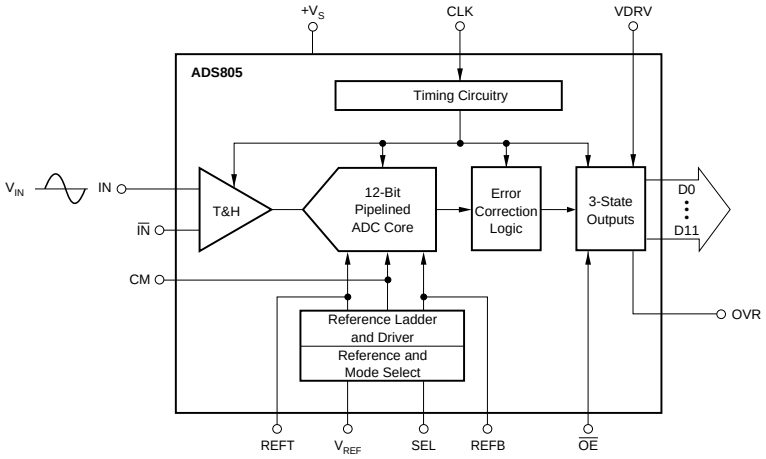
- **STUDIO CAMERAS**
- **IF AND BASEBAND DIGITIZATION**
- **COPIERS**
- **TEST INSTRUMENTATION**

DESCRIPTION

The ADS805 is a 20MHz, high dynamic range, 12-bit, pipelined Analog-to-Digital Converter ADC. This converter includes a high-bandwidth track-and-hold that gives excellent spurious performance up to and beyond the Nyquist rate. This high-bandwidth, linear track-and-hold minimizes harmonics and has low jitter, leading to excellent Signal-to-Noise Ratio (SNR) performance. The ADS805 is also pin-compatible with the 10MHz ADS804 and the 5MHz ADS803.

The ADS805 provides an internal reference or an external reference can be used. The ADS805 can be programmed for a 2Vp-p input range which is the easiest to drive with a single op amp and provides the best spurious performance. Alternatively, the 5Vp-p input range can be used for the lowest input-referred noise of 0.09LSBs rms giving superior imaging performance. There is also the capability to set the input range between 2Vp-p and 5Vp-p, either single-ended or differential. The ADS805 also provides an over-range flag that indicates when the input signal has exceeded the converter's full-scale range. This flag can also be used to reduce the gain of the front end signal conditioning circuitry.

The ADS805 employs digital error techniques to provide excellent differential linearity for demanding imaging applications. Its low distortion and high SNR give the extra margin needed for communications, medical imaging, video, and test instrumentation applications. The ADS805 is available in an SSOP-28 package.



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

ABSOLUTE MAXIMUM RATINGS[1]

$+V_S$	+6V
Analog Input	–0.3V to $(+V_S)$ + 0.3V
Logic Input	–0.3V to $(+V_S)$ + 0.3V
Case Temperature	+100°C
Junction Temperature	+150°C
Storage Temperature	+150°C

NOTE: (1) Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum conditions for extended periods may affect device reliability.

ELECTROSTATIC DISCHARGE SENSITIVITY

This integrated circuit can be damaged by ESD. Texas Instruments recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

PACKAGE/ORDERING INFORMATION

PRODUCT	PACKAGE-LEAD	PACKAGE DESIGNATOR[1]	SPECIFIED TEMPERATURE RANGE	PACKAGE MARKING	ORDERING NUMBER[2]	TRANSPORT MEDIA, QUANTITY
ADS805	SSOP-28	DB	–40°C to +85°C	ADS805E	ADS805E	Rails, 48
"	"	"	"	"	ADS805E/1K	Tape and Reel, 1000

NOTE: (1) For the most current specifications and package information, refer to our web site at www.ti.com.

ELECTRICAL CHARACTERISTICS

At T_A = full specified temperature range, V_S = +5V, specified input range = 1.5V to 3.5V, and single-ended input and sampling rate = 20MHz, unless otherwise specified.

PARAMETER	CONDITIONS	ADS805E MIN	ADS805E TYP	ADS805E MAX	UNITS
RESOLUTION			12 Bits Tested		
SPECIFIED TEMPERATURE RANGE			–40 to +85		°C
CONVERSION CHARACTERISTICS					
Sample Rate		10k		20M	Samples/s
Data Latency			6		Clk Cycles
ANALOG INPUT					
Standard Single-Ended Input Range		1.5		3.5	V
Optional Single-Ended Input Range		0		5	V
Standard Common-Mode Voltage			2.5		V
Standard Optional Common-Mode Voltage			1		V
Input Capacitance			20		pF
Analog Input Bandwidth	–3dBFS Input		270		MHz
DYNAMIC CHARACTERISTICS					
Differential Linearity Error (Largest Code Error)					
f = 500kHz			±0.25	±0.75	LSB
No Missing Codes			Tested		
Spurious-Free Dynamic Range[1]					
f = 9.8MHz		65	74		dBFS[2]
2-Tone Intermodulation Distortion[3]					
f = 7.7MHz and 7.9MHz (–7dB each tone)			–70		dBc
Signal-to-Noise Ratio (SNR)					
f = 9.8MHz		63	68		dBFS
Signal-to-(Noise + Distortion) (SINAD)					
f = 9.8MHz		62	66		dBFS
Effective Number of Bits at 9.8MHz[4]			10.7		Bits
Input Referred Noise	0V to 5V Input		0.09		LSBs rms
	1.5V to 3.5V Input		0.23		LSBs rms
Integral Nonlinearity Error					
f = 500kHz			±1	±2	LSB
Aperture Delay Time			3		ns
Aperture Jitter			4		ps rms
Over-Voltage Recovery Time	1.5x FS Input		2		ns
Full-Scale Step Acquisition Time			20		ns

NOTES: (1) Spurious-Free Dynamic Range refers to the magnitude of the largest harmonic. (2) dBFS means dB relative to full-scale. (3) 2-tone intermodulation distortion is referred to the largest fundamental tone. This number will be 6dB higher if it is referred to the magnitude of the 2-tone fundamental envelope. (4) Effective number of bits (ENOB) is defined by (SINAD – 1.76)/6.02. (5) Internal 50kΩ pull-down resistor. (6) Includes internal reference. (7) Excludes internal reference.

ELECTRICAL CHARACTERISTICS (Cont.)

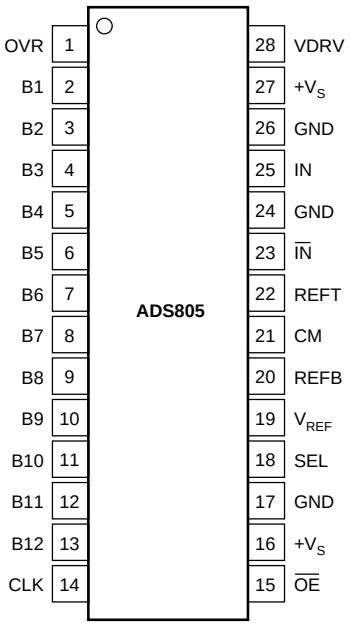
At T_A = full specified temperature range, V_S = +5V, specified input range = 1.5V to 3.5V, and single-ended input and sampling rate = 20MHz, unless otherwise specified.

PARAMETER	CONDITIONS	ADS805E MIN	ADS805E TYP	ADS805E MAX	UNITS
DIGITAL INPUTS					
Logic Family			CMOS Compatible		
Convert Command	Start Conversion		Rising Edge of Convert Clock		
High Level Input Current (V_{IN} = 5V)[5]				±100	µA
Low Level Input Current (V_{IN} = 0V)				10	µA
High Level Input Voltage		+3.5			V
Low Level Input Voltage				+1.0	V
Input Capacitance			5		pF
DIGITAL OUTPUTS					
Logic Family			CMOS/TTL Compatible		
Logic Coding			Straight Offset Binary		
Low Output Voltage	(I_{OL} = 50µA)			0.1	V
Low Output Voltage	(I_{OL} = 1.6mA)			0.4	V
High Output Voltage	(I_{OH} = 50µA)	+4.5			V
High Output Voltage	(I_{OH} = 0.5mA)	+2.4			V
3-State Enable Time	$\overline{OE}$ = L		20	40	ns
3-State Disable Time	$\overline{OE}$ = H		2	10	ns
Output Capacitance			5		pF
ACCURACY (5Vp-p Input Range)	f_S = 2.5MHz				
Zero-Error (Referred to –FS)	At 25°C		0.3	±1.5	%FS
Zero-Error Drift (Referred to –FS)			±5		ppm/°C
Gain Error[6]	At 25°C		0.7	±2.0	%FS
Gain Error Drift[6]			±18		ppm/°C
Gain Error[7]	At 25°C		0.2	±1.5	%FS
Gain Error Drift[7]			±10		ppm/°C
Power-Supply Rejection of Gain	ΔV_S = ±5%	60	70		dB
Reference Input Resistance			1.6		kΩ
Internal Voltage Reference Tolerance (V_{REF} = 2.5V)	At 25°C			±35	mV
Internal Voltage Reference Tolerance (V_{REF} = 1.0V)	At 25°C			±14	mV
POWER-SUPPLY REQUIREMENTS					
Supply Voltage: $+V_S$	Operating	+4.75	+5.0	+5.25	V
Supply Current: $+I_S$	Operating		60	69	mA
Power Dissipation	Operating		300	345	mW
Thermal Resistance, θ_{JA}					
SSOP-28		50		°C/W	

NOTES: (1) Spurious-Free Dynamic Range refers to the magnitude of the largest harmonic. (2) dBFS means dB relative to full-scale. (3) 2-tone intermodulation distortion is referred to the largest fundamental tone. This number will be 6dB higher if it is referred to the magnitude of the 2-tone fundamental envelope. (4) Effective number of bits (ENOB) is defined by (SINAD – 1.76)/6.02. (5) Internal 50kΩ pull-down resistor. (6) Includes internal reference. (7) Excludes internal reference.

PIN CONFIGURATION

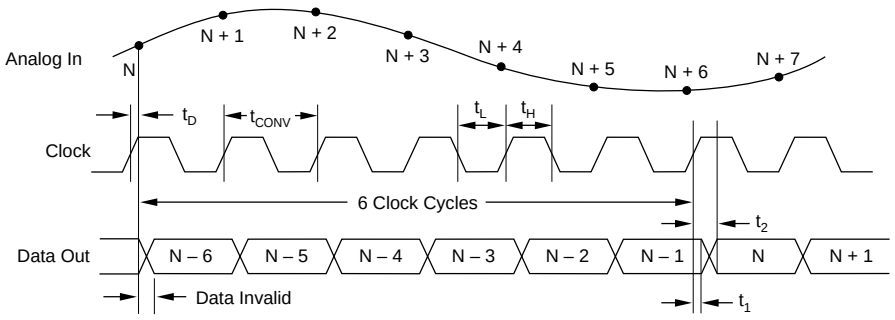
Top View **SSOP**



PIN DESCRIPTIONS

PIN	DESIGNATOR	DESCRIPTION
1	OVR	Over-Range Indicator
2	B1	Data Bit 1 (D11) (MSB)
3	B2	Data Bit 2 (D10)
4	B3	Data Bit 3 (D9)
5	B4	Data Bit 4 (D8)
6	B5	Data Bit 5 (D7)
7	B6	Data Bit 6 (D6)
8	B7	Data Bit 7 (D5)
9	B8	Data Bit 8 (D4)
10	B9	Data Bit 9 (D3)
11	B10	Data Bit 10 (D2)
12	B11	Data Bit 11 (D1)
13	B12	Data Bit 12 (D0) (LSB)
14	CLK	Convert Clock Input
15	$\overline{OE}$	Output Enable. H = High Impedance State. L = LOW or floating, normal operation (internal pull-down resistor).
16	$+V_S$	+5V Supply
17	GND	Ground
18	SEL	Input Range Select
19	V_{REF}	Reference Voltage Select
20	REFB	Bottom Reference
21	CM	Common-Mode Voltage
22	REFT	Top Reference
23	$\overline{IN}$	Complementary Analog Input
24	GND	Ground
25	IN	Analog Input (+)
26	GND	Ground
27	$+V_S$	+5V Supply
28	VDRV	Output Driver Voltage

TIMING DIAGRAM

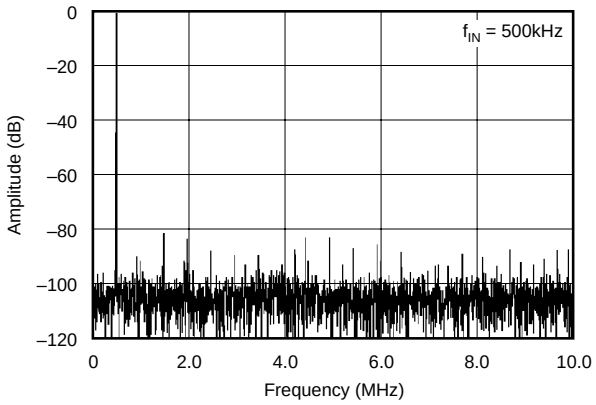


SYMBOL	DESCRIPTION	MIN	TYP	MAX	UNITS
t_{CONV}	Convert Clock Period	50		100µs	ns
t_L	Clock Pulse LOW	24	25		ns
t_H	Clock Pulse HIGH	24	25		ns
t_D	Aperture Delay		3		ns
t_1	Data Hold Time, C_L = 0pF	3.9			ns
t_2	New Data Delay Time, C_L = 15pF max			12	ns

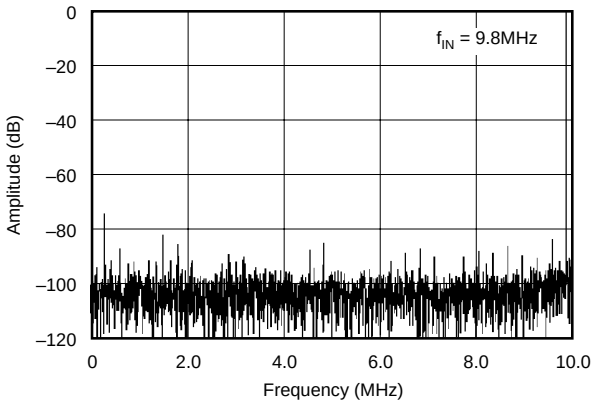
TYPICAL CHARACTERISITCS

At T_A = full specified temperature range, V_S = +5V, specified single-ended input range = 1.5V to 3.5V, and sampling rate = 20MHz, unless otherwise specified.

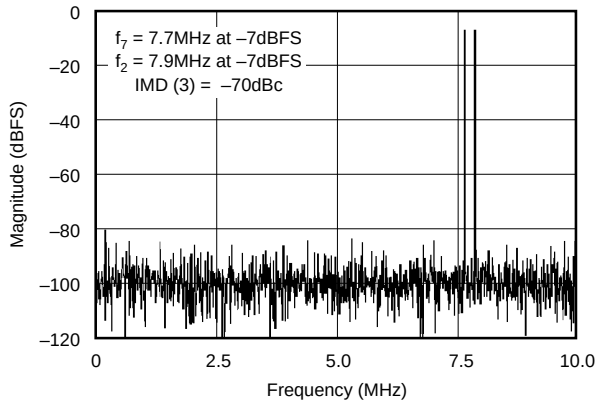
SPECTRAL PERFORMANCE



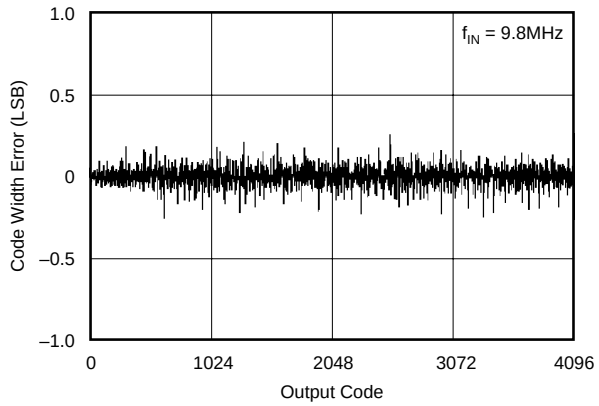
SPECTRAL PERFORMANCE



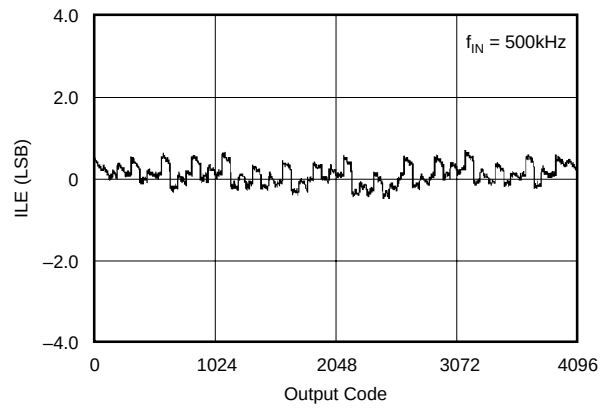
2-TONE INTERMODULATION DISTORTION



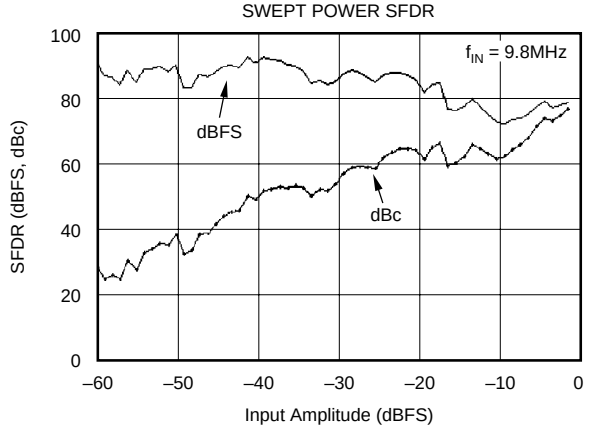
DIFFERENTIAL LINEARITY ERROR



INTEGRAL LINEARITY ERROR



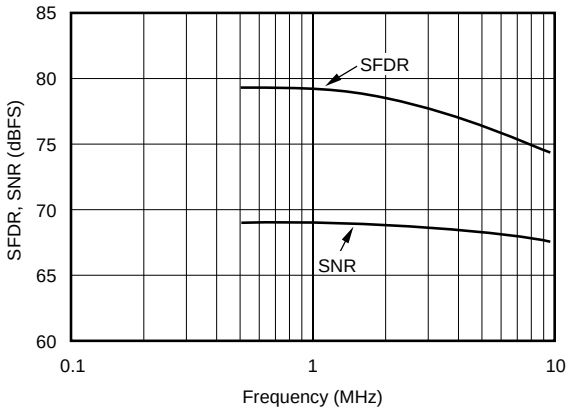
SWEPT POWER SFDR



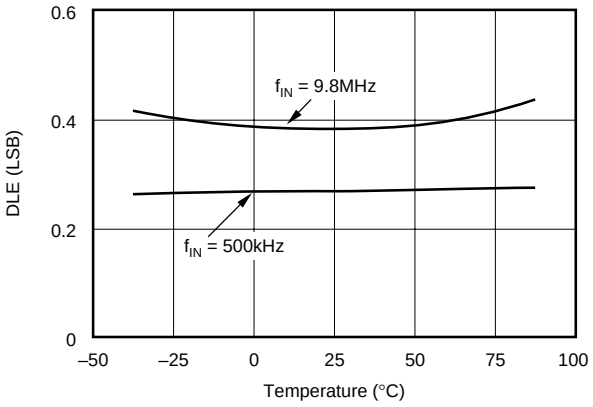
TYPICAL CHARACTERISITCS (Cont.)

At T_A = full specified temperature range, V_S = +5V, specified single-ended input range = 1.5V to 3.5V, and sampling rate = 20MHz, unless otherwise specified.

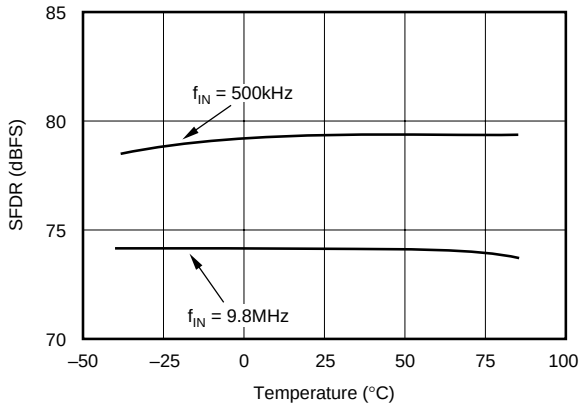
DYNAMIC PERFORMANCE vs INPUT FREQUENCY



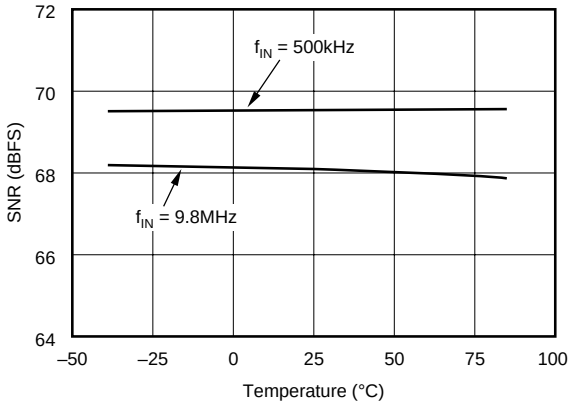
DIFFERENTIAL LINEARITY ERROR vs TEMPERATURE



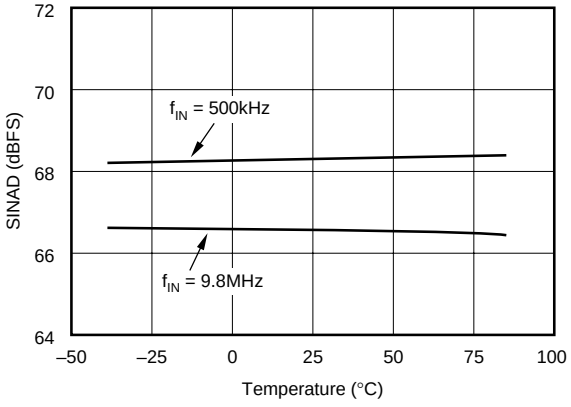
SPURIOUS-FREE DYNAMIC RANGE vs TEMPERATURE



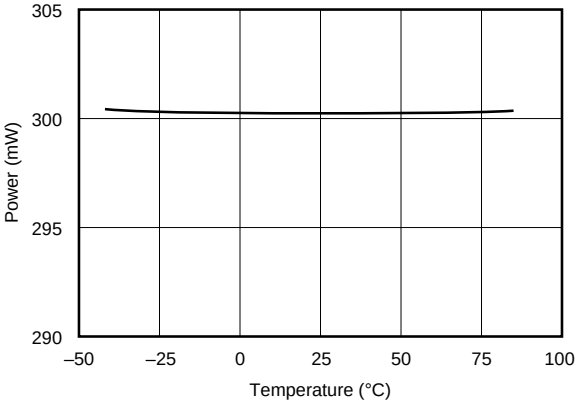
SIGNAL-TO-NOISE RATIO vs TEMPERATURE



SIGNAL-TO-(NOISE + DISTORTION) vs TEMPERATURE



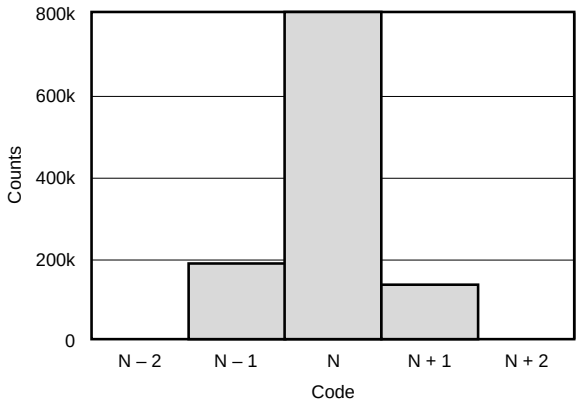
POWER DISSIPATION vs TEMPERATURE



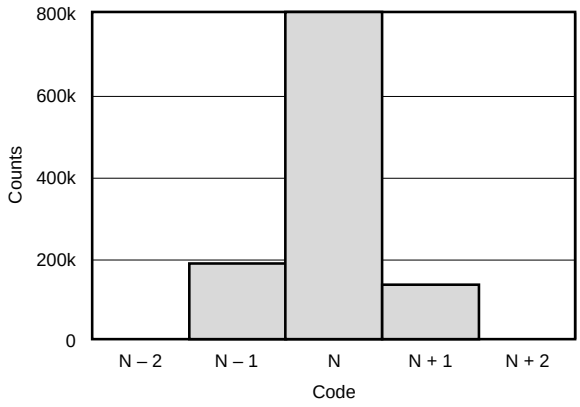
TYPICAL CHARACTERISITCS (Cont.)

At T_A = full specified temperature range, V_S = +5V, specified single-ended input range = 1.5V to 3.5V, and sampling rate = 20MHz, unless otherwise specified.

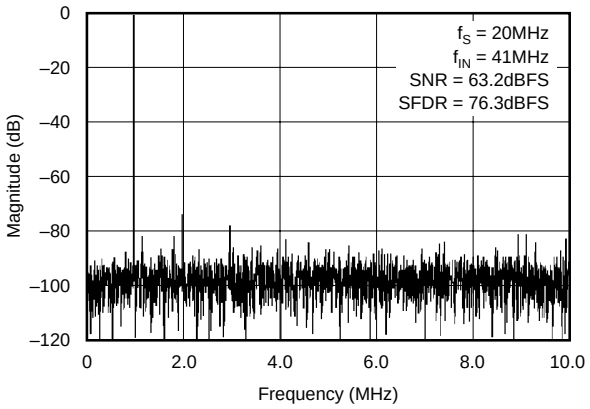
OUTPUT NOISE HISTOGRAM
(DC Input)



OUTPUT NOISE HISTOGRAM
(DC Input, V_{IN} = 5Vp-p Range)



UNDERSAMPLING
(Differential Input, 2Vp-p)



APPLICATION INFORMATION

DRIVING THE ANALOG INPUT

The ADS805 allows its analog inputs to be driven either single-ended or differentially. The focus of the following discussion is on the single-ended configuration. Typically, its implementation is easier to achieve and the rated specifications for the ADS805 are characterized using the single-ended mode of operation.

AC-COUPLED INPUT CONFIGURATION

Given in Figure 1 is the circuit example of the most common interface configuration for the ADS805. With the V_{REF} pin connected to the SEL pin, the full-scale input range is defined to be 2Vp-p. This signal is ac-coupled in single-ended form to the ADS805 using the low distortion voltage-feedback amplifier OPA642. As is generally necessary for single-supply components, operating the ADS805 with a full-scale input signal swing requires a level-shift of the amplifier's zero centered analog signal to comply with the ADC's input range requirements. Using a DC-blocking capacitor between the output of the driving amplifier and the converter's input, a simple level-shifting scheme can be implemented. In this configuration, the top and bottom references (REFT, REFB) provide an output voltage of +3V and +2V, respectively. Here, two resistor pairs (2 • 2kΩ) are used to create a common-mode voltage of approximately +2.5V to bias the inputs of the ADS805 (IN, $\overline{IN}$) to the required DC voltage.

An advantage of ac-coupling is that the driving amplifier still operates with a ground-based signal swing. This will keep the distortion performance at its optimum since the signal swing stays within the linear region of the op amp and sufficient headroom to the supply rails can be maintained. Consider using the inverting gain configuration to eliminate CMR induced errors of the amplifier. The addition of a small series resistor (R_S) between the output of the op amp and the input of the ADS805 will be beneficial in almost all interface configurations. This will decouple the op amp's output from the capacitive load and avoid gain peaking, which can result in increased noise. For best spurious and distortion performance, the resistor value should be kept below 100Ω. Furthermore, the series resistor, together with the 100pF capacitor, establish a passive low-pass filter, limiting the bandwidth for the wideband noise, thus helping improve the signal-to-noise performance.

DC-COUPLED WITHOUT LEVEL SHIFT

In some applications the analog input signal may already be biased at a level which complies with the selected input range and reference level of the ADS805. In this case, it is only necessary to provide an adequately low source impedance to the selected input, IN or $\overline{IN}$. Always consider wideband op amps since their output impedance will stay low over a wide range of frequencies.

DC-COUPLED WITH LEVEL SHIFT

Several applications may require that the bandwidth of the signal path include DC, in which case the signal has to be DC-coupled to the ADC. In order to accomplish this, the interface circuit has to provide a DC-level shift. The circuit presented in Figure 2 utilizes the single-supply, current-feedback op amp OPA681 (A1), to sum the ground-centered input signal with a required DC offset. The ADS805 typically operates with a +2.5V common-mode voltage, which is established with resistors R_3 and R_4 and connected to the $\overline{IN}$ input of the converter. Amplifier A1 operates in inverting configuration. Here, resistors R_1 and R_2 set the DC-bias level for A1. Because of the op amp's noise gain of +2V/V, assuming $R_F = R_{IN}$, the DC offset voltage applied to its noninverting input has to be divided down to +1.25V, resulting in a DC output voltage of +2.5V. DC voltage differences between the IN and $\overline{IN}$ inputs of the ADS805 effectively will produce an offset, which can be corrected for by adjusting the values of resistors R_1 and R_2. The bias current of the op amp may also result in an undesired

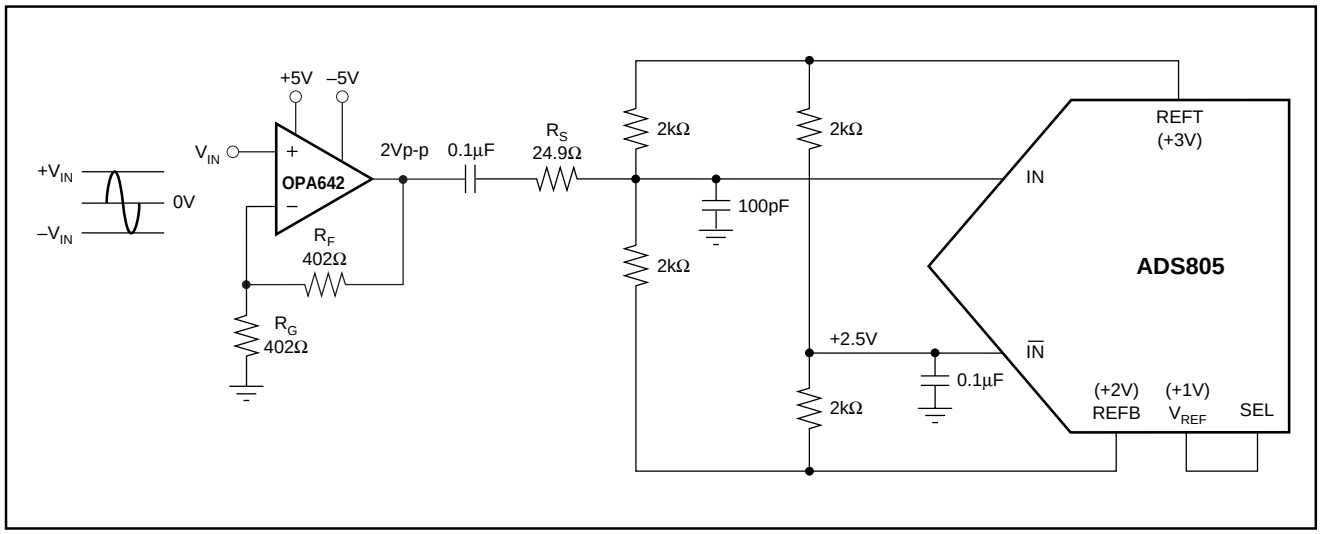


FIGURE 1. AC-Coupled Input Configuration for 2Vp-p Input Swing and Common-Mode Voltage at +2.5V Derived from Internal Top and Bottom Reference.

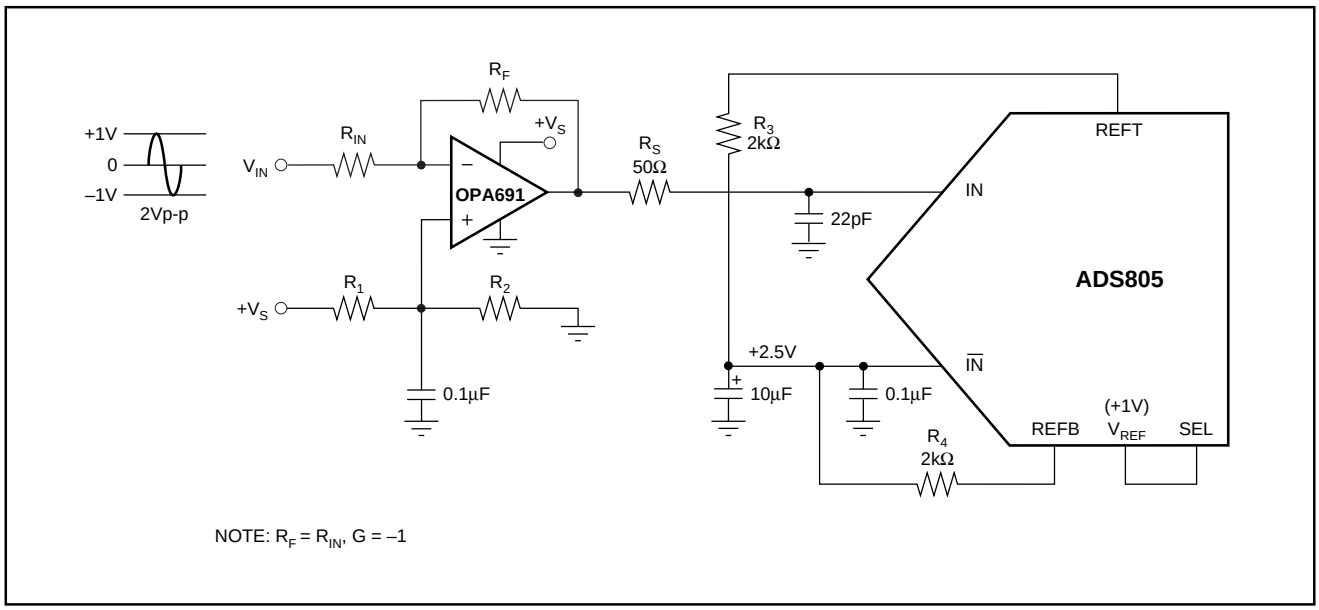


FIGURE 2. DC-Coupled, Single-Ended Input Configuration with DC-level Shift.

offset. The selection criteria for an appropriate op amp should include the input bias current, output voltage swing, distortion, and noise specification. Note that in this example the overall signal phase is inverted. To reestablish the original signal polarity, it is always possible to interchange the IN and $\overline{IN}$ connections.

SINGLE-ENDED-TO-DIFFERENTIAL CONFIGURATION (TRANSFORMER-COUPLED)

In order to select the best suited interface circuit for the ADS805, the performance requirements must be known. If an ac-coupled input is needed for a particular application, the next step is to determine the method of applying the signal; either single-ended or differentially. The differential input configuration may provide a noticeable advantage of achieving good SFDR performance based on the fact that, in the differential mode, the signal swing can be reduced to half of the swing required for single-ended drive. Secondly, by driving the ADS805 differentially, the even-order harmonics will be reduced. Figure 3 shows the schematic for the suggested transformer-coupled interface circuit. The resistor across the secondary side (R_T) should be set to get an input impedance match (e.g., $R_T = n^2 \bullet R_G$).

One application example that will benefit from the differential input configuration is the digitization of IF signals. The wide track-and-hold input bandwidth makes the ADS805 well suited for IF down conversion in both narrow and wideband applications. The ADS805 maintains excellent dynamic performance in multiple Nyquist regions covering a variety of IF frequencies (see the Typical Characteristics). Using the ADS805 for direct IF conversion eliminates the need of an analog mixer along with subsequent functions like amplifiers and filters, thus reducing system cost and complexity.

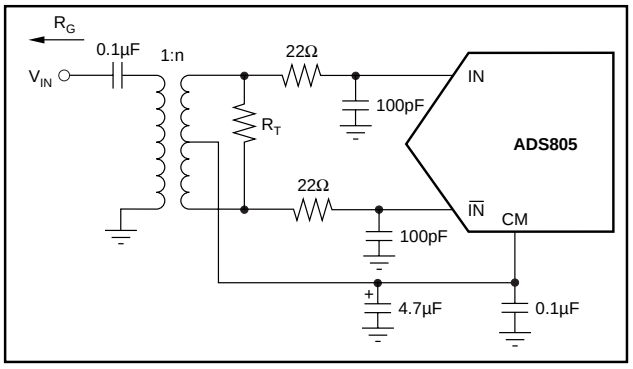


FIGURE 3. Transformer-Coupled Input.

REFERENCE OPERATION

Integrated into the ADS805 is a bandgap reference circuit including logic that provides either a +1V or +2.5V reference output, by simply selecting the corresponding pin-strap configuration. Different reference voltages can be generated by the use of two external resistors, which will set a different gain for the internal reference buffer. For more design flexibility, the internal reference can be shut off and an external reference voltage used. Table I provides an overview of the possible reference options and pin configurations.

MODE	INPUT FULL-SCALE RANGE	REQUIRED V_{REF}	CONNECT	TO
Internal	2Vp-p	+1V	SEL	V_{REF}
Internal	5Vp-p	+2.5V	SEL	GND
Internal	2V ≤ FSR < 5V FSR = 2 • V_{REF}	1V < V_{REF} < 2.5V $V_{REF} = 1 + (R_1/R_2)$	R_1 R_2	V_{REF} and SEL SEL and Gnd
External	1V < FSR < 5V	0.5V < V_{REF} < 2.5V	SEL V_{REF}	$+V_S$ Ext. V_{REF}

TABLE I. Selected Reference Configuration Examples.

A simple model of the internal reference circuit is shown in Figure 4. The internal blocks are a 1V-bandgap voltage reference, buffer, the resistive reference ladder and the drivers for the top and bottom reference which supply the necessary current to the internal nodes. As shown, the output of the buffer appears at the V_{REF} pin. The full-scale input span of the ADS805 is determined by the voltage at V_{REF}, according to Equation 1:

$$\text{Full-Scale Input Span} = 2 \bullet V_{REF} \quad (1)$$

Note that the current drive capability of this amplifier is limited to approximately 1mA and should not be used to drive low loads. The programmable reference circuit is controlled by the voltage applied to the select pin (SEL). Refer to Table I for an overview.

The top reference (REFT) and the bottom reference (REFB) are brought out mainly for external bypassing. For proper operation with all reference configurations, it is necessary to provide solid bypassing to the reference pins in order to keep the clock feedthrough to a minimum. Figure 5 shows the recommended decoupling network.

In addition, the Common-Mode Voltage (CMV) may be used as a reference level to provide the appropriate offset for the driving circuitry. However, care must be taken not to appreciably load this node, which is not buffered and has a high impedance. An alternate method of generating a common-mode voltage is given in Figure 6. Here, two external precision resistors (tolerance 1% or better) are located between the top and bottom reference pins. The common-mode level will appear at the midpoint. The output buffers of the top and bottom reference are designed to supply approximately 2mA of output current.

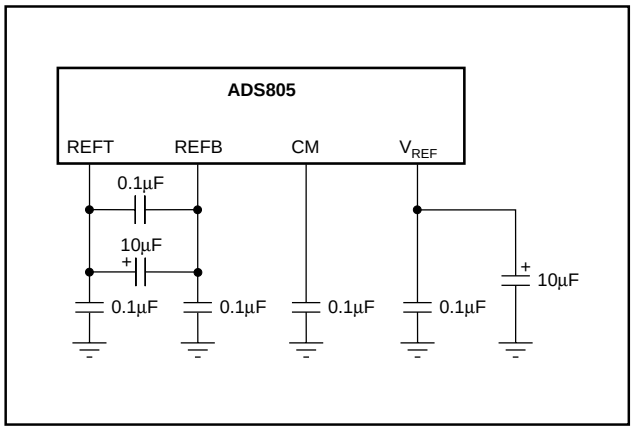


FIGURE 5. Recommended Reference Bypassing Scheme.

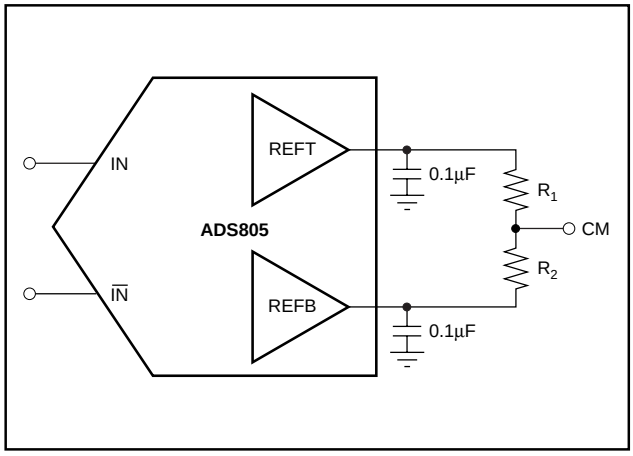


FIGURE 6. Alternative Circuit to Generate Common-Mode Voltage.

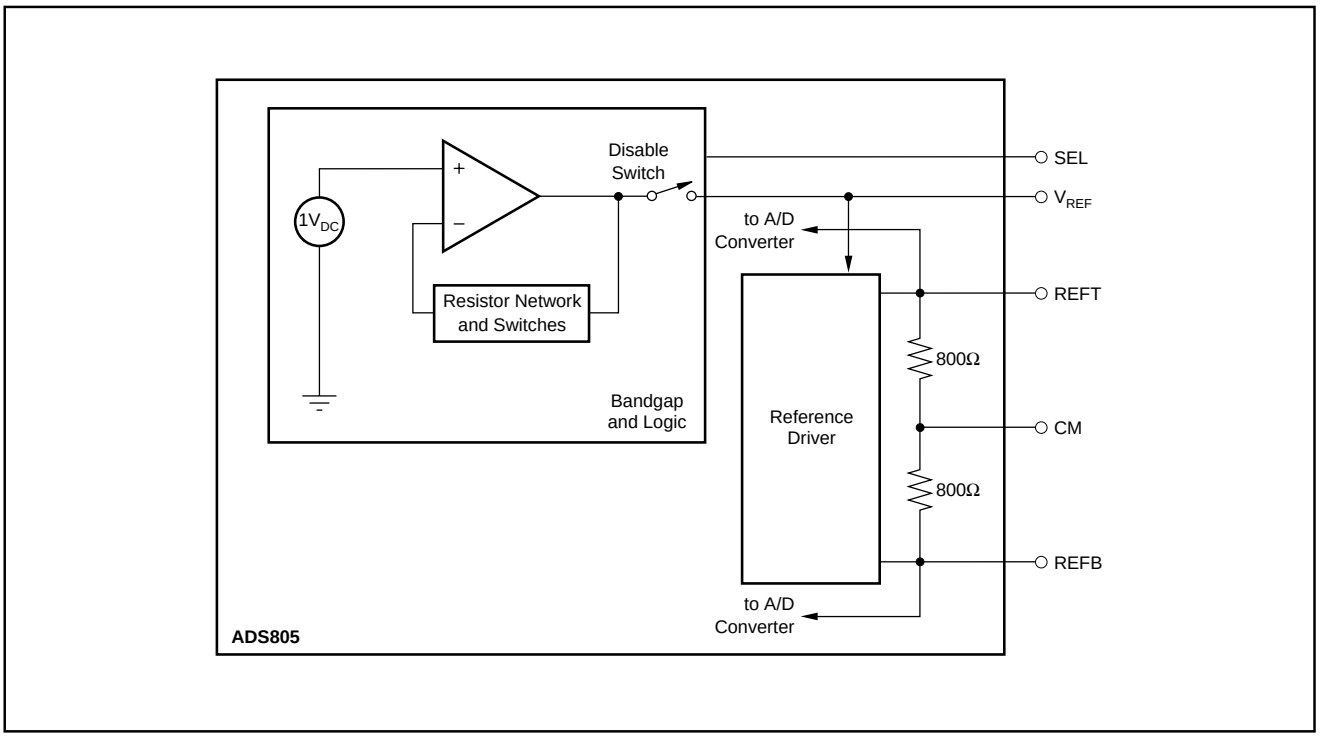


FIGURE 4. Equivalent Reference Circuit.

SELECTING THE INPUT RANGE AND REFERENCE

Figures 7 through 9 show a selection of circuits for the most common input ranges when using the internal reference of the ADS805. All examples are for single-ended input and operate with a nominal common-mode voltage of +2.5V.

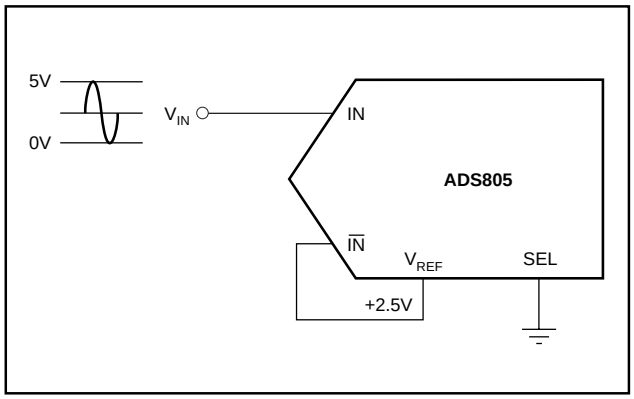


FIGURE 7. Internal Reference with 0V to 5V Input Range.

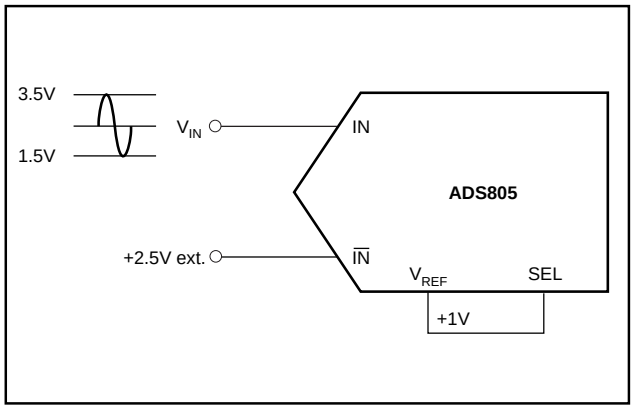


FIGURE 8. Internal Reference with 1.5V to 3.5V Input Range.

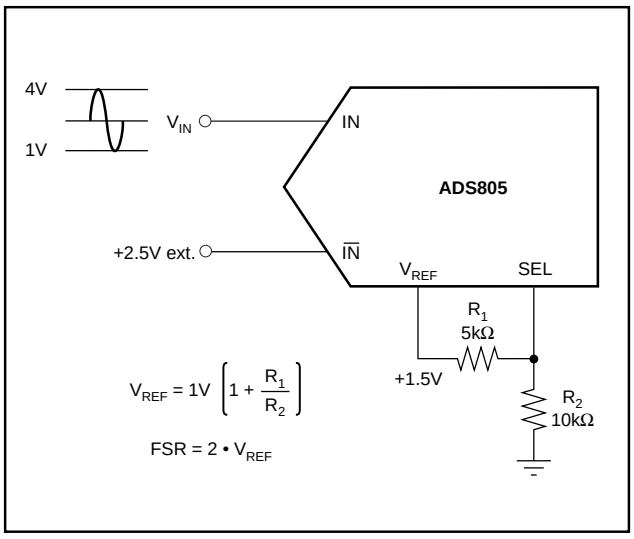


FIGURE 9. Internal Reference with 1V to 4V Input Range.

EXTERNAL REFERENCE OPERATION

Depending on the application requirements, it might be advantageous to operate the ADS805 with an external reference. This may improve the DC accuracy if the external reference circuitry is superior in its drift and accuracy. To use the ADS805 with an external reference, the user must disable the internal reference, as shown in Figure 10. By connecting the SEL pin to $+V_S$, the internal logic will shut down the internal reference. At the same time, the output of the internal reference buffer is disconnected from the V_{REF} pin, which now must be driven with the external reference. Note that a similar bypassing scheme should be maintained as described for the internal reference operation.

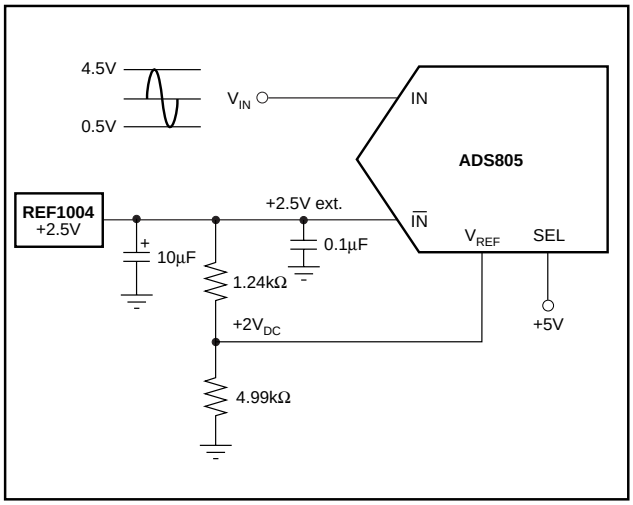


FIGURE 10. External Reference, Input Range 0.5V to 4.5V (4Vp-p), with +2.5V Common-Mode Voltage.

DIGITAL INPUTS AND OUTPUTS

Over-Range (OVR)

One feature of the ADS805 is its 'Over-Range' (OVR) digital output. This pin can be used to monitor any out-of-range condition, which occurs every time the applied analog input voltage exceeds the input range (set by V_{REF}). The OVR output is LOW when the input voltage is within the defined input range. It becomes HIGH when the input voltage is beyond the input range. This is the case when the input voltage is either below the bottom reference voltage or above the top reference voltage. OVR will remain active until the analog input returns to its normal signal range and another conversion is completed. Using the MSB and its complement in conjunction with OVR, a simple decode logic can be built that detects the over-range and under-range conditions, (see Figure 11). It should be noted that OVR is a digital output which is updated along with the bit information corresponding to the particular sampling incidence of the analog signal. Therefore, the OVR data is subject to the same pipeline delay (latency) as the digital data.

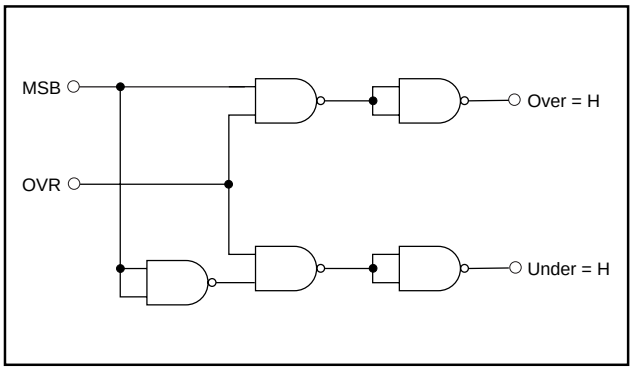


FIGURE 11. External Logic for Decoding Under-Range and Over-Range Conditions.

CLOCK INPUT REQUIREMENTS

Clock jitter is critical to the SNR performance of high-speed, high-resolution ADCs. It leads to aperture jitter (t_A) which adds noise to the signal being converted. The ADS805 samples the input signal on the rising edge of the CLK input. Therefore, this edge should have the lowest possible jitter. The jitter noise contribution to total SNR is given by Equation 2. If this value is near your system requirements, input clock jitter must be reduced.

$$\text{Jitter SNR} = 20\log\frac{1}{2\pi f_{IN} t_A}\ \text{rms signal to rms noise} \quad (2)$$

Where: f_{IN} is Input Signal Frequency,

t_A is rms Clock Jitter

Particularly in undersampling applications, special consideration should be given to clock jitter. The clock input should be treated as an analog input in order to achieve the highest level of performance. Any overshoot or undershoot of the clock signal may cause degradation of the performance. When digitizing at high sampling rates, the clock should have a 50% duty cycle ($t_H = t_L$), along with fast rise-and-fall times of 2ns or less.

DIGITAL OUTPUTS

The digital outputs of the ADS805 are designed to be compatible with both high-speed TTL and CMOS logic families. The driver stage for the digital outputs is supplied through a separate supply pin, VDRV, which is not connected to the analog supply pins. By adjusting the voltage on VDRV, the digital output levels will vary respectively. Therefore, it is possible to operate the ADS805 on a +5V analog supply while interfacing the digital outputs to 3V-logic with the VDRV pin tied to the +3V digital supply.

It is recommended to keep the capacitive loading on the data lines as low as possible (≤ 15pF). Larger capacitive loads demand higher charging currents as the outputs are changing. Those high-current surges can feed back to the analog portion of the ADS805 and influence the performance. If necessary, external buffers or latches may be used which provide the added benefit of isolating the ADS805 from any digital noise activities on the bus coupling back high-frequency noise. In addition, resistors in series with each data line may help maintain the ac performance of the ADS805. Their use depends on the capacitive loading seen by the converter. Values in the range of 100Ω to 200Ω will limit the instantaneous current the output stage has to provide for recharging the parasitic capacitances, as the output levels change from LOW to HIGH or HIGH to LOW.

GROUNDING AND DECOUPLING

Proper grounding and bypassing, short lead length, and the use of ground planes are particularly important for high-frequency designs. Multilayer PC boards are recommended for best performance since they offer distinct advantages like minimizing ground impedance, separation of signal layers by ground layers, etc. It is recommended that the analog and digital ground pins of the ADS805 be joined together at the IC and be connected only to the analog ground of the system.

The ADS805 has analog and digital supply pins, however the converter should be treated as an analog component and all supply pins should be powered by the analog supply. This will ensure the most consistent results, since digital supply lines often carry high levels of noise that would otherwise be coupled into the converter and degrade the achievable performance.

Because of the pipeline architecture, the converter also generates high-frequency current transients and noise that are fed back into the supply and reference lines. This requires that the supply and reference pins be sufficiently bypassed. Figure 12 shows the recommended decoupling scheme for the analog supplies. In most cases, 0.1µF ceramic chip capacitors are adequate to keep the impedance low over a wide frequency range. Their effectiveness largely depends on the proximity to the individual supply pin. Therefore, they should be located as close to the supply pins as possible. In addition, a larger size bipolar capacitor (1µF to 22µF) should be placed on the PC board in close proximity to the converter circuit.

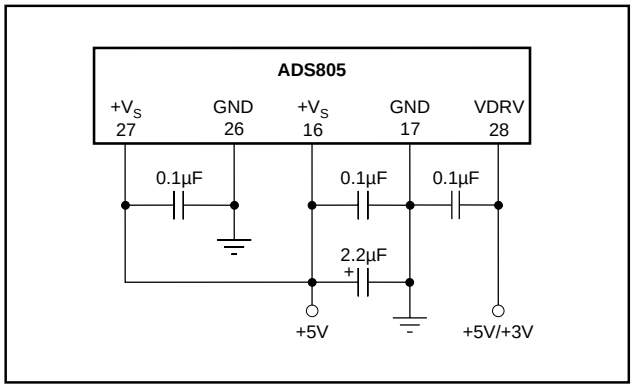


FIGURE 12. Recommended Bypassing for Analog Supply Pins.

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
ADS805E	ACTIVE	SSOP	DB	28	50	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 85	ADS805E	Samples
ADS805E/1K	ACTIVE	SSOP	DB	28	1000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 85	ADS805E	Samples
ADS805U	OBSOLETE	SOIC	DW	28		TBD	Call TI	Call TI	-40 to 85		
ADS805U/1K	OBSOLETE	SOIC	DW	28		TBD	Call TI	Call TI	-40 to 85		

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. - The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

(4) There may be additional marking, which relates to the logo, the lot trace code information, or the environmental category on the device.

(5) Multiple Device Markings will be inside parentheses. Only one Device Marking contained in parentheses and separated by a "~" will appear on a device. If a line is indented then it is a continuation of the previous line and the two combined represent the entire Device Marking for that device.

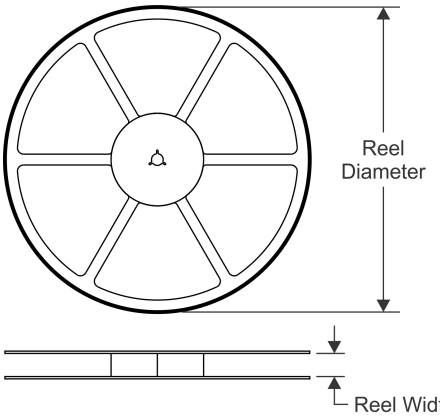
(6) Lead/Ball Finish - Orderable Devices may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead/Ball Finish values may wrap to two lines if the finish value exceeds the maximum column width.

continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals.
TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

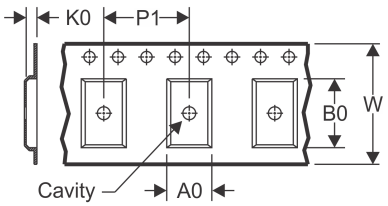
In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.

TAPE AND REEL INFORMATION

REEL DIMENSIONS

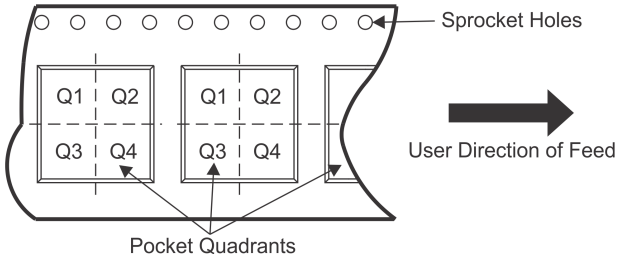


TAPE DIMENSIONS



A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers

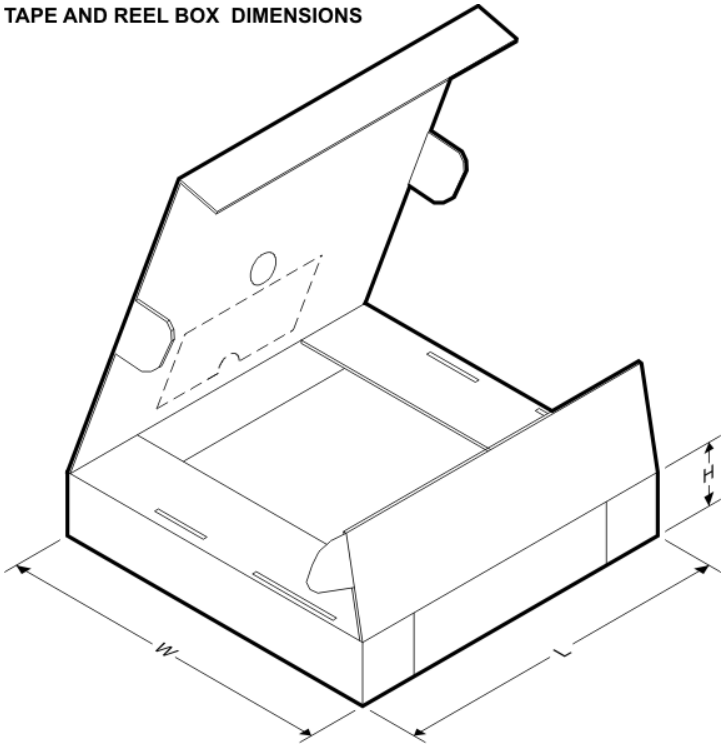
QUADRANT ASSIGNMENTS FOR PIN 1 ORIENTATION IN TAPE



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
ADS805E/1K	SSOP	DB	28	1000	330.0	16.4	8.1	10.4	2.5	12.0	16.0	Q1

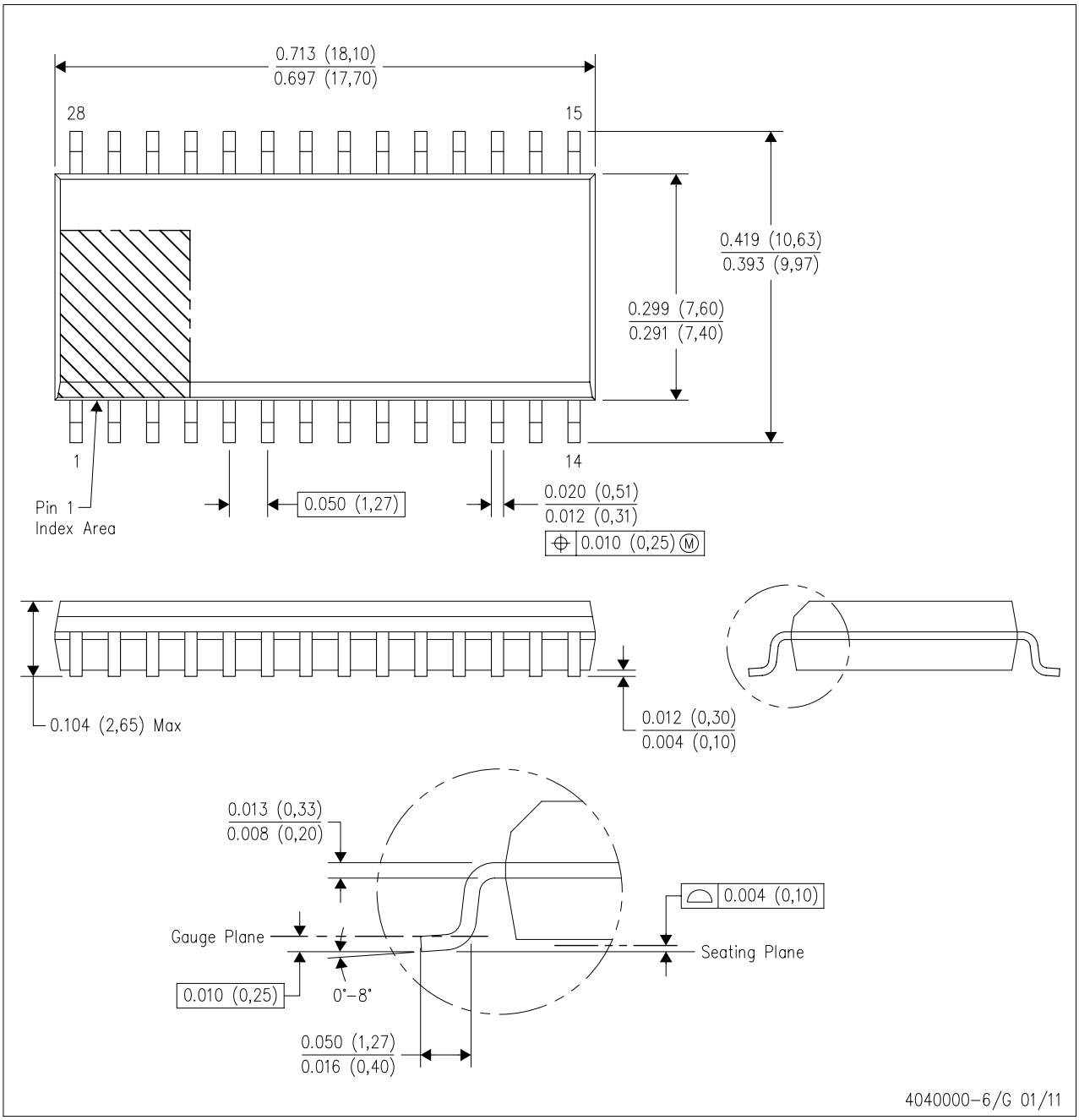
TAPE AND REEL BOX DIMENSIONS



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
ADS805E/1K	SSOP	DB	28	1000	367.0	367.0	38.0

DW (R–PDSO–G28) PLASTIC SMALL OUTLINE

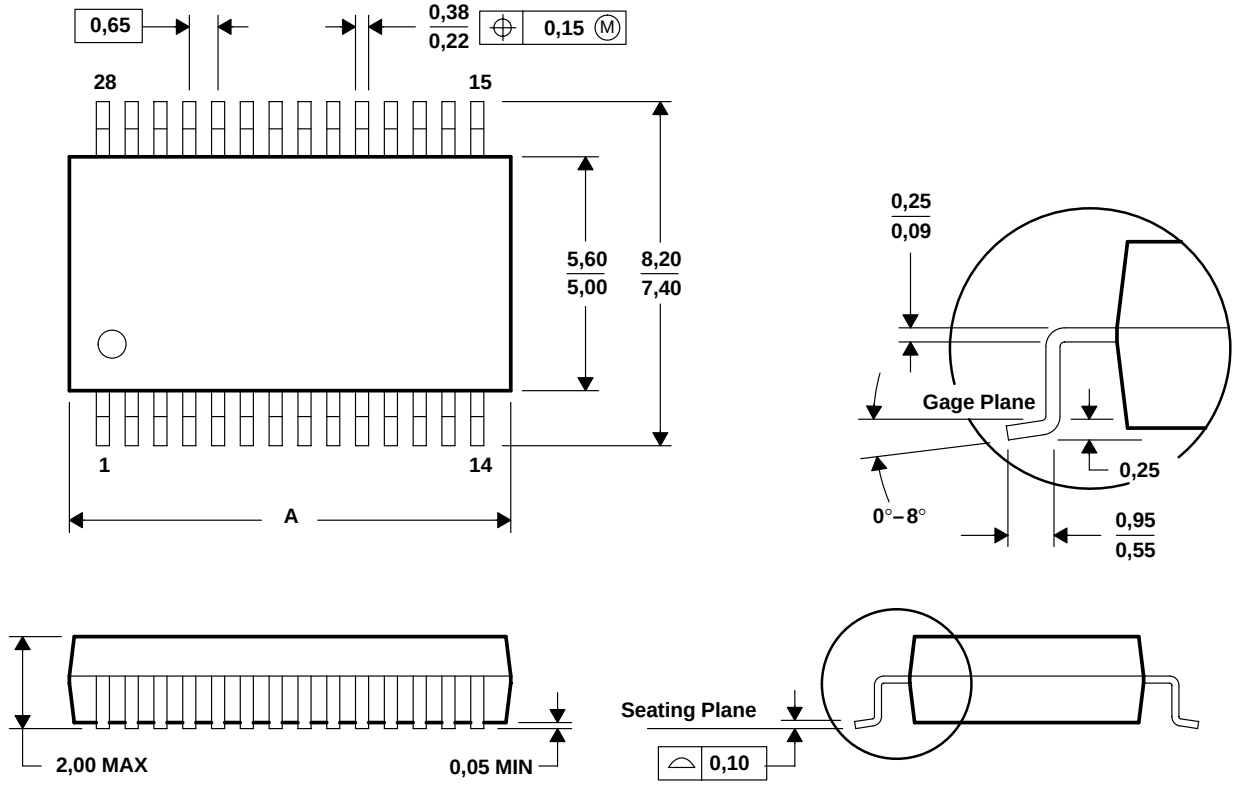


NOTES:
A. All linear dimensions are in inches (millimeters). Dimensioning and tolerancing per ASME Y14.5M–1994.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion not to exceed 0.006 (0,15).
D. Falls within JEDEC MS–013 variation AE.

MSSO002E – JANUARY 1995 – REVISED DECEMBER 2001

DB (R-PDSO-G)** **PLASTIC SMALL-OUTLINE**

28 PINS SHOWN



DIM \ PINS **	14	16	20	24	28	30	38
A MAX	6,50	6,50	7,50	8,50	10,50	10,50	12,90
A MIN	5,90	5,90	6,90	7,90	9,90	9,90	12,30

4040065 /E 12/01

NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion not to exceed 0,15.
D. Falls within JEDEC MO-150

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Dual Programmable Gain Amplifiers with Serial Digital Interface

FEATURES

- **2 Channels with Independent Gain Control**
  LTC6912-1: (0, 1, 2, 5, 10, 20, 50, and 100V/V)
  LTC6912-2: (0, 1, 2, 4, 8, 16, 32, and 64V/V)
- **Offset Voltage = 2mV Max (–40°C to 85°C)**
- **Channel-to-Channel Gain Matching of 0.1dB Max**
- **3-Wire SPI™ Interface**
- **Extended Gain-Bandwidth at High Gains**
- Wired-OR Outputs Possible (2:1 Analog MUX Function)
- Low Power Hardware Shutdown (GN-16 Only, 2µA Max at 2.7V)
- Rail-to-Rail Input Range
- Rail-to-Rail Output Swing
- Single or Dual Supply: 2.7V to 10.5V Total
- Input Noise: 12.6nV/√Hz
- Total System Dynamic Range to 115dB
- 16-Pin GN (SSOP) or 12-Pin DFN Package Options

APPLICATIONS

- Data Acquisition Systems
- Dynamic Gain Changing
- Automatic Ranging Circuits
- Automatic Gain Control

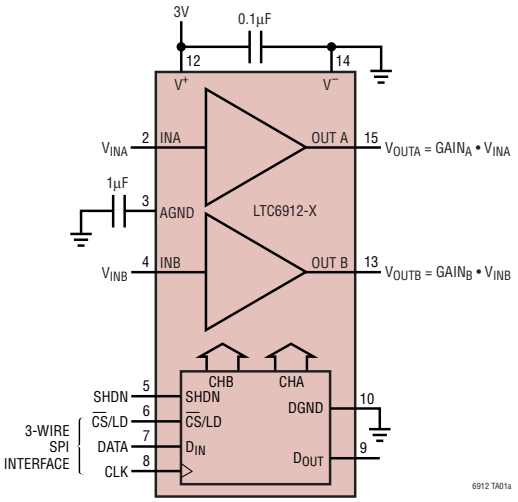
DESCRIPTION

The LTC®6912 is a family of dual channel, low noise, digitally programmable gain amplifiers (PGA) that are easy to use and occupy very little PC board space. The gains for both channels are independently programmable using a 3-wire SPI interface to select voltage gains of 0, 1, 2, 5, 10, 20, 50, and 100V/V (LTC6912-1); and 0, 1, 2, 4, 8, 16, 32, and 64V/V (LTC6912-2). All gains are inverting.

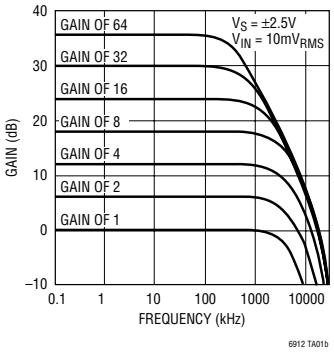
The LTC6912 family consists of 2 matched amplifiers with rail-to-rail outputs. When operated with unity gain, they will also process rail-to-rail input signals. A half-supply reference generated internally at the AGND pin supports single power supply applications. Operating from single or split supplies from 2.7V to 10.5V total, the LTC6912-X family is offered in tiny SSOP and DFN-12 Packages.

TYPICAL APPLICATION

A Dual, Matched Low Noise PGA (16-Lead SSOP Package)



LTC6912-2
Frequency Response

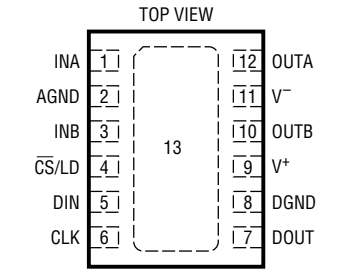


ABSOLUTE MAXIMUM RATINGS

(Note 1)

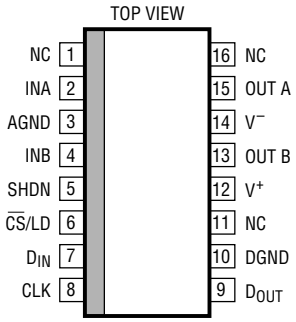
Total Supply Voltage (V^+ to V^-) 11V
Input Current ..... ±10mA
Operating Temperature Range (Note 2)
LTC6912C-1, LTC6912C-2 .....–40°C to 85°C
LTC6912I-1, LTC6912I-2 .....–40°C to 85°C
LTC6912H-1, LTC6912H-2
(GN-16 Only) .....–40°C to 125°C
Specified Temperature Range (Note 3)
LTC6912C-1, LTC6912C-2 .....–40°C to 85°C
LTC6912I-1, LTC6912I-2 .....–40°C to 85°C
LTC6912H-1, LTC6912H-2
(GN-16 Only) .....–40°C to 125°C
Storage Temperature Range .....–65°C to 150°C
UE Package .....–65°C to 125°C
Lead Temperature (Soldering, 10sec)..... 300°C

PACKAGE/ORDER INFORMATION



UE12 PACKAGE
12-LEAD (4mm × 3mm) PLASTIC DFN
EXPOSED PAD IS CONNECTED TO V^- (PIN 13),
MUST BE SOLDERED TO PCB

T_{JMAX} = 125°C, θ_{JA} = 160°C/W



GN PACKAGE
16-LEAD NARROW PLASTIC SSOP

T_{JMAX} = 150°C, θ_{JA} = 120°C/W

ORDER PART NUMBER	DFN PART* MARKING	ORDER PART NUMBER	GN PART MARKING
LTC6912CDE-1	69121	LTC6912CGN-1	69121
LTC6912IDE-1	69121	LTC6912IGN-1	6912I1
LTC6912CDE-2	69122	LTC6912HGN-1	6912H1
LTC6912IDE-2	69122	LTC6912CGN-2	69122
		LTC6912IGN-2	6912I2
		LTC6912HGN-2	6912H2

Order Options Tape and Reel: Add #TR Lead Free: Add #PBF Lead Free Tape and Reel: Add #TRPBF
Lead Free Part Marking: http://www.linear.com/leadfree/

Consult LTC Marketing for parts specified with wider operating temperature ranges. *The temperature grade is identified by a label on the shipping container.

GAIN SETTINGS AND PROPERTIES

Table 1. LTC6912-1 GAIN SETTINGS AND PROPERTIES

UPPER/LOWER NIBBLE				NOMINAL VOLTAGE GAIN		MAXIMUM LINEAR INPUT RANGE (V_{P-P})				
Q7 Q3	Q6 Q2	Q5 Q1	Q4 Q0	Volts/Volt	dB	Dual 5V Supply	Single 5V Supply	Single 3V Supply	NOMINAL INPUT IMPEDANCE (kΩ)	NOMINAL OUTPUT IMPEDANCE (Ω)
0	0	0	0	0	–120	10	5	3	(Open)	0.4
0	0	0	1	–1	0	10	5	3	10	0.7
0	0	1	0	–2	6	5	2.5	1.5	5	3.4
0	0	1	1	–5	14	2	1	0.6	2	3.4
0	1	0	0	–10	20	1	0.5	0.3	1	3.4
0	1	0	1	–20	26	0.5	0.25	0.15	1	6.4
0	1	1	0	–50	34	0.2	0.1	0.06	1	15
0	1	1	1	–100	40	0.1	0.05	0.03	1	30
1	0	X	X	0	–120	10	5	3	(Open)	(Open)
1	1	X	X	Not Used (Note 11)					Not Used	

Table 2. LTC6912-2 GAIN SETTINGS AND PROPERTIES

UPPER/LOWER NIBBLE				NOMINAL VOLTAGE GAIN		MAXIMUM LINEAR INPUT RANGE (V_{P-P})				
Q7 Q3	Q6 Q2	Q5 Q1	Q4 Q0	Volts/Volt	dB	Dual 5V Supply	Single 5V Supply	Single 3V Supply	NOMINAL INPUT IMPEDANCE (kΩ)	NOMINAL OUTPUT IMPEDANCE (Ω)
0	0	0	0	0	–120	10	5	3	(Open)	0.4
0	0	0	1	–1	0	10	5	3	10	0.7
0	0	1	0	–2	6	5	2.5	1.5	5	3.4
0	0	1	1	–4	12	2.5	1.25	0.75	2.5	3.4
0	1	0	0	–8	18.1	1.25	0.625	0.375	1.25	3.4
0	1	0	1	–16	24.1	0.625	0.3125	0.188	1.25	6.4
0	1	1	0	–32	30.1	0.3125	0.156	0.094	1.25	15
0	1	1	1	–64	36.1	0.156	0.078	0.047	1.25	30
1	0	X	X	0	–120	10	5	3	(Open)	(Open)
1	1	X	X	Not Used (Note 11)					Not Used	

ELECTRICAL CHARACTERISTICS

The ● denotes the specifications that apply over the full operating temperature range, otherwise specifications are at T_A = 25°C. V_S = 5V, AGND = 2.5V, Gain = 1, R_L = 10k to midsupply point, unless otherwise noted.

PARAMETER	CONDITIONS		C, I GRADES MIN	C, I GRADES TYP	C, I GRADES MAX	H GRADE MIN	H GRADE TYP	H GRADE MAX	UNITS
Specifications for Both the LTC6912-1 and the LTC6912-2									
Total Supply Voltage (V_S)		●	2.7		10.5	2.7		10.5	V
Supply Current per Channel	Both Amplifiers Active (Gain = 1)								
	V_S = 2.7V, V_{INA} = V_{INB} = V_{AGND}	●		1.75	2.75		1.75	3.0	mA
	V_S = 5V, V_{INA} = V_{INB} = V_{AGND}	●		2.0	3.0		2.0	3.25	mA
	V_S = ±5V, V_{INA} = V_{INB} = 0V	●		2.25	3.5		2.25	3.75	mA
Supply Current per Channel (Software Shutdown)	Both Amplifiers Inactive (State 1000)								
	V_S = 2.7V, V_{INA} = V_{INB} = V_{AGND}	●		150	255		150	280	µA
	V_S = 5V, V_{INA} = V_{INB} = V_{AGND}	●		200	325		200	350	µA
	V_S = ±5V, V_{INA} = V_{INB} = 0V	●		265	750		265	750	µA
Total-Supply Current (Hardware Shutdown, GN-16 Package Only)	V_S = 2.7V, V_{SHDN} = 2.43V	●		0.3	2		0.3	5	µA
	V_S = 5V, V_{SHDN} = 4.5V	●		3.6	10		3.6	10	µA
	V_S = ±5V, V_{SHDN} = 4.5V	●		20	50		20	50	µA
Output Voltage Swing LOW (Note 4)	V_S = 2.7V, R_L = 10k Tied to Midsupply Point	●		12	30		12	35	mV
	V_S = 2.7V, R_L = 500Ω Tied to Midsupply Point	●		60	110		50	125	mV
	V_S = 5V, R_L = 10k Tied to Midsupply Point	●		20	40		20	45	mV
	V_S = 5V, R_L = 500Ω Tied to Midsupply Point	●		100	170		90	190	mV
	V_S = ±5V, R_L = 10k Tied to 0V	●		30	50		30	60	mV
	V_S = ±5V, R_L = 500Ω Tied to 0V	●		190	260		80	290	mV
Output Voltage Swing HIGH (Note 4)	V_S = 2.7V, R_L = 10k Tied to Midsupply Point	●		10	20		10	25	mV
	V_S = 2.7V, R_L = 500Ω Tied to Midsupply Point	●		50	80		50	90	mV
	V_S = 5V, R_L = 10k Tied to Midsupply Point	●		15	30		15	35	mV
	V_S = 5V, R_L = 500Ω Tied to Midsupply Point	●		90	160		80	175	mV
	V_S = ±5V, R_L = 10k Tied to 0V	●		20	40		20	45	mV
	V_S = ±5V, R_L = 500Ω Tied to 0V	●		180	250		180	270	mV
Output Short-Circuit Current (Note 5)	V_S = 2.7V	●		±27			±27		mA
	V_S = ±5V	●		±35			±35		mA
AGND Open-Circuit Voltage (GN-16 Package Only)	V_S = Single 5V Supply, V_{SHDN} = 0.5V	●	2.45	2.5	2.55	2.45	2.5	2.55	V
	V_S = Single 5V Supply, V_{SHDN} = 4.5V			2.65			2.65		V
AGND (Common Mode) Input Voltage Range	V_S = Single 2.7V Supply	●	0.55		1.6	0.55		1.6	V
	V_S = Single 5V Supply	●	0.75		3.65	0.75		3.65	V
	V_S = ±5V	●	–4.3		3.2	–4.3		3.2	V
AGND Rejection (i.e., Common Mode Rejection or CMRR)	V_S = 2.7V, V_{AGND} = 1.1V to 1.6V	●	55	80		50	80		dB
	V_S = ±5V, V_{AGND} = –2.5V to 2.5V	●	55	75		50	75		dB
Power Supply Rejection Ratio (PSRR)	V_S =2.7V to ±5V	●	60	80		57	80		dB
Slew Rate	Gain = 1								
	V_S = 5V, V_{OUTA} = V_{OUTB} = 1.1V to 3.9V			12			12		V/µs
	V_S = ±5V, V_{OUTA} = V_{OUTB} = ±1.4V			16			16		V/µs
	Gain = 10 (–1), Gain = 8 (–2)								
	V_S = 5V, V_{OUTA} = V_{OUTB} = 1.1V to 3.9V			20			20		V/µs
	V_S = ±5V, V_{OUTA} = V_{OUTB} = ±1.4V			26			26		V/µs
Signal Attenuation at Gain = 0 Setting	Gain = 0 (Digital Inputs 0000), f = 200kHz	●		–120			–120		dB
Signal Attenuation in Software Shutdown	(State = 1000)	●		–120			–120		dB

ELECTRICAL CHARACTERISTICS

The ● denotes the specifications that apply over the full operating temperature range, otherwise specifications are at T_A = 25°C. V_S = 5V, AGND = 2.5V, Gain = 1, R_L = 10k to midsupply point, unless otherwise noted.

PARAMETER	CONDITIONS		C, I GRADES MIN	C, I GRADES TYP	C, I GRADES MAX	H GRADE MIN	H GRADE TYP	H GRADE MAX	UNITS
Specifications for Both the LTC6912-1 and the LTC6912-2									
SHDN Input High Voltage (GN-16 Package Only)	V_S = Single 2.7V	●	2.43			2.43			V
	V_S = Single 5V	●	4.5			4.5			V
	V_S = ±5V	●	4.5			4.5			V
SHDN Input Low Voltage (GN-16 Package Only)	V_S = Single 2.7V	●			0.27			0.27	V
	V_S = Single 5V	●			0.5			0.5	V
	V_S = ±5V	●			0.5			0.5	V
SHDN Pin 5, Input High Current (GN-16 Package Only)	V_S = Single 2.7V			0.2			0.2		µA
	V_S = Single 5V			1			1		µA
	V_S = ±5V			1			1		µA
SHDN Pin 5, Input Low Current (GN-16 Package Only)	V_S = Single 2.7V			0.2			0.2		µA
	V_S = Single 5V			1			1		µA
	V_S = ±5V			1			1		µA
Specifications for the LTC6912-1 ONLY									
Voltage Gain (Note 6)	V_S = 2.7V, Gain = 1, R_L = 10k	●	–0.07	0	0.07	–0.08	0	0.07	dB
	V_S = 2.7V, Gain = 1, R_L = 500Ω	●	–0.11	–0.02	0.07	–0.13	–0.02	0.07	dB
	V_S = 2.7V, Gain = 2, R_L = 10k	●	5.94	6.01	6.08	5.93	6.01	6.08	dB
	V_S = 2.7V, Gain = 5, R_L = 10k	●	13.85	13.95	14.05	13.8	13.95	14.05	dB
	V_S = 2.7V, Gain = 10, R_L =10k	●	19.7	19.93	20.1	19.65	19.93	20.1	dB
	V_S = 2.7V, Gain = 10, R_L = 500Ω	●	19.55	19.85	20.05	19.35	19.85	20.05	dB
	V_S = 2.7V, Gain = 20, R_L = 10k	●	25.75	25.94	26.1	25.65	25.94	26.1	dB
	V_S = 2.7V, Gain = 50, R_L = 10k	●	33.5	33.8	34.05	33.40	33.8	34.05	dB
	V_S = 2.7V, Gain = 100, R_L = 10k	●	39.2	39.6	40.0	39.0	39.6	40.0	dB
	V_S = 2.7V, Gain = 100, R_L = 500Ω	●	37.3	38.9	39.7	36.20	38.9	39.7	dB
	V_S = 5V, Gain = 1, R_L = 10k	●	–0.08	0.01	0.08	–0.09	0.01	0.08	dB
	V_S = 5V, Gain = 1, R_L = 500Ω	●	–0.11	–0.01	0.07	–0.13	–0.01	0.07	dB
	V_S = 5V, Gain = 2, R_L = 10k	●	5.95	6.02	6.09	5.94	6.02	6.09	dB
	V_S = 5V, Gain = 5, R_L = 10k	●	13.8	13.96	14.1	13.78	13.96	14.1	dB
	V_S = 5V, Gain = 10, R_L = 10k	●	19.8	19.94	20.1	19.75	19.94	20.1	dB
	V_S = 5V, Gain = 10, R_L = 500Ω	●	19.6	19.87	20.1	19.45	19.87	20.1	dB
	V_S = 5V, Gain = 20, R_L = 10k	●	25.78	25.94	26.08	25.75	25.94	26.08	dB
	V_S = 5V, Gain = 50, R_L = 10k	●	33.5	33.84	34.1	33.4	33.84	34.1	dB
	V_S = 5V, Gain = 100, R_L = 10k	●	39.3	39.7	40.1	39.1	39.7	40.1	dB
	V_S = 5V, Gain = 100, R_L = 500Ω	●	37.75	39.2	39.85	36.6	39.2	39.85	dB
	V_S = ±5V, Gain = 1, R_L = 10k	●	–0.06	0.01	0.08	–0.07	0.01	0.08	dB
	V_S = ±5V, Gain = 1, R_L = 500Ω	●	–0.10	0	0.08	–0.11	0	0.08	dB
	V_S = ±5V, Gain = 2, R_L = 10k	●	5.95	6.02	6.09	5.94	6.02	6.09	dB
	V_S = ±5V, Gain = 5, R_L = 10k	●	13.8	13.96	14.1	13.79	13.96	14.1	dB
	V_S = ±5V, Gain = 10, R_L = 10k	●	19.78	19.94	20.08	19.75	19.94	20.08	dB
	V_S = ±5V, Gain = 10, R_L = 500Ω	●	19.68	19.91	20.05	19.58	19.91	20.05	dB
	V_S = ±5V, Gain = 20, R_L = 10k	●	25.78	25.95	26.08	25.73	25.95	26.08	dB
	V_S = ±5V, Gain = 50, R_L = 10k	●	33.65	33.87	34.05	33.60	33.87	34.05	dB
	V_S = ±5V, Gain = 100, R_L = 10k	●	39.4	39.8	40.2	39.25	39.8	40.2	dB
	V_S = ±5V, Gain = 100, R_L = 500Ω	●	38.6	39.5	39.9	37.6	39.5	39.9	dB

ELECTRICAL CHARACTERISTICS

The ● denotes the specifications that apply over the full operating temperature range, otherwise specifications are at T_A = 25°C. V_S = 5V, AGND = 2.5V, Gain = 1, R_L = 10k to midsupply point, unless otherwise noted.

PARAMETER	CONDITIONS		C, I GRADES MIN	C, I GRADES TYP	C, I GRADES MAX	H GRADE MIN	H GRADE TYP	H GRADE MAX	UNITS
Specifications for the LTC6912-1 ONLY									
Channel-to-Channel Voltage Gain Match (Note 6)	V_S = 2.7V, Gain = 1, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 2.7V, Gain = 1, R_L = 500Ω	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 2.7V, Gain = 2, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 2.7V, Gain = 5, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 2.7V, Gain = 10, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 2.7V, Gain = 10, R_L = 500Ω	●	–0.15	±0.02	0.15	–0.2	±0.02	0.2	dB
	V_S = 2.7V, Gain = 20, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 2.7V, Gain = 50, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 2.7V, Gain = 100, R_L = 10k	●	–0.2	±0.02	0.2	–0.2	±0.02	0.2	dB
	V_S = 2.7V, Gain = 100, R_L = 500Ω	●	–1.0	±0.02	1.0	–1.5	±0.02	1.5	dB
	V_S = 5V, Gain = 1, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 5V, Gain = 1, R_L = 500Ω	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 5V, Gain = 2, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 5V, Gain = 5, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 5V, Gain = 10, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 5V, Gain = 10, R_L = 500Ω	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 5V, Gain = 20, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 5V, Gain = 50, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 5V, Gain = 100, R_L = 10k	●	–0.2	±0.02	0.2	–0.2	±0.02	0.2	dB
	V_S = 5V, Gain = 100, R_L = 500Ω	●	–0.8	±0.02	0.8	–1.2	±0.02	1.2	dB
	V_S = ±5V, Gain = 1, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = ±5V, Gain = 1, R_L = 500Ω	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = ±5V, Gain = 2, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = ±5V, Gain = 5, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = ±5V, Gain = 10, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = ±5V, Gain = 10, R_L = 500Ω	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = ±5V, Gain = 20, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = ±5V, Gain = 50, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = ±5V, Gain = 100, R_L = 10k	●	–0.2	±0.02	0.2	–0.2	±0.02	0.2	dB
	V_S = ±5V, Gain = 100, R_L = 500Ω	●	–0.6	±0.02	0.6	–0.9	±0.02	0.9	dB
Gain Temperature Coefficient (Note 6)	V_S = 5V, Gain = 1, R_L = OPEN			2			2		ppm/°C
	V_S = 5V, Gain = 2, R_L = OPEN			–1.5			–1.5		ppm/°C
	V_S = 5V, Gain = 5, R_L = OPEN			–11			–11		ppm/°C
	V_S = 5V, Gain = 10, R_L = OPEN			–30			–30		ppm/°C
	V_S = 5V, Gain = 20, R_L = OPEN			–40			–40		ppm/°C
	V_S = 5V, Gain = 50, R_L = OPEN			–70			–70		ppm/°C
	V_S = 5V, Gain = 100, R_L = OPEN			–140			–140		ppm/°C
Channel-to-Channel Gain Temperature Coefficient Match (Gain Specified in dB's) (Note 6)	V_S = 5V, Gain = 1, R_L = OPEN			1			1		ppm/°C
	V_S = 5V, Gain = 2, R_L = OPEN			1			1		ppm/°C
	V_S = 5V, Gain = 5, R_L = OPEN			0.2			0.2		ppm/°C
	V_S = 5V, Gain = 10, R_L = OPEN			–1			–1		ppm/°C
	V_S = 5V, Gain = 20, R_L = OPEN			–1			–1		ppm/°C
	V_S = 5V, Gain = 50, R_L = OPEN			–3			–3		ppm/°C
	V_S = 5V, Gain = 100, R_L = OPEN			–3			–3		ppm/°C
Channel-to-Channel Isolation (Note 7)	f = 200kHz,								
	V_S = 5V, Gain = 1, R_L = 10k			113			113		dB
	V_S = 5V, Gain = 10, R_L = 10k			108			108		dB
	V_S = 5V, Gain = 100, R_L = 10k			89			89		dB

ELECTRICAL CHARACTERISTICS

The ● denotes the specifications that apply over the full operating temperature range, otherwise specifications are at T_A = 25°C. V_S = 5V, AGND = 2.5V, Gain = 1, R_L = 10k to midsupply point, unless otherwise noted.

PARAMETER	CONDITIONS		C, I SUFFIXES MIN	C, I SUFFIXES TYP	C, I SUFFIXES MAX	H SUFFIX MIN	H SUFFIX TYP	H SUFFIX MAX	UNITS
Specifications for the LTC6912-1 ONLY									
Offset Voltage Magnitude (Internal Op-Amp, Note 8)	Gain = 1	●		0.125	2		0.125	3.5	mV
Offset Voltage Magnitude Referred to INA or INB Pins (Note 8)	Gain = 1	●		0.25	3.5		0.25	6.5	mV
	Gain = 10	●		0.14	2		0.14	4	mV
Input Offset Voltage Drift, Internal Op Amp				6			10		µV/°C
DC Input Resistance at INA or INB Pins (Note 9)	DC V_{INA} or V_{INB} = 0V								
	Gain = 0	●		>10			>10		MΩ
	State = 8, Software Shutdown	●		>10			>10		MΩ
	Gain = 1	●		10			10		kΩ
	Gain = 2	●		5			5		kΩ
	Gain = 5	●		2			2		kΩ
	Gain > 5	●		1			1		kΩ
DC Input Resistance Drift at INA or INB Pins (Note 9)	Gain = 1			85			95		ppm/°C
	Gain = 2			90			100		ppm/°C
	Gain = 5			100			110		ppm/°C
	Gain = 10			120			130		ppm/°C
	Gain = 20			130			140		ppm/°C
	Gain = 50			150			160		ppm/°C
	Gain = 100			190			200		ppm/°C
DC Input Resistance Match R_{INA}-R_{INB}	Gain = 1	●		10			10		Ω
	Gain = 2	●		5			5		Ω
	Gain = 5	●		5			5		Ω
	Gain > 5	●		5			5		Ω
DC Small Signal Output Resistance at OUT A or OUT B Pins	DC V_{INA} or V_{INB} = 0V								
	Gain = 0			0.4			0.4		Ω
	Gain = 1			0.7			0.7		Ω
	Gain = 2			1.0			1.0		Ω
	Gain = 5			1.9			1.9		Ω
	Gain = 10			3.4			3.4		Ω
	Gain = 20			6.4			6.4		Ω
	Gain = 50			15			15		Ω
	Gain = 100			30			30		Ω
	State = 8, Software Shutdown	●		>1			>1		MΩ
Gain Bandwidth Product	Gain = 100	●	18	33	50	16	33	50	MHz
Wideband Noise (Referred to Input)	f = 1kHz to 200kHz								
	Gain = 0 (Output Noise only)			8.9			8.9		$µV_{RMS}$
	Gain = 1			15.6			15.6		$µV_{RMS}$
	Gain = 2			11.1			11.1		$µV_{RMS}$
	Gain = 5			8.3			8.3		$µV_{RMS}$
	Gain = 10			7.4			7.4		$µV_{RMS}$
	Gain = 20			7.0			7.0		$µV_{RMS}$
	Gain = 50			6.7			6.7		$µV_{RMS}$
	Gain = 100			6.3			6.3		$µV_{RMS}$

ELECTRICAL CHARACTERISTICS

The ● denotes the specifications that apply over the full operating temperature range, otherwise specifications are at T_A = 25°C. V_S = 5V, AGND = 2.5V, Gain = 1, R_L = 10k to midsupply point, unless otherwise noted.

PARAMETER	CONDITIONS		C, I GRADES MIN	C, I GRADES TYP	C, I GRADES MAX	H GRADE MIN	H GRADE TYP	H GRADE MAX	UNITS
Specifications for the LTC6912-1 ONLY									
Voltage Noise Density (Referred to Input)	f = 50kHz								
	Gain = 1			35.6			35.6		nV/√Hz
	Gain = 2			24.8			24.8		nV/√Hz
	Gain = 5			19.1			19.1		nV/√Hz
	Gain = 10			16.7			16.7		nV/√Hz
	Gain = 20			16			16		nV/√Hz
	Gain = 50			15.4			15.4		nV/√Hz
	Gain = 100			15.1			15.1		nV/√Hz
Total Harmonic Distortion	Gain = 10, f_{IN} = 10kHz, V_{OUT} = 1V_{RMS}			–90			–90		dB
				0.003			0.003		%
	Gain = 10, f_{IN} = 100kHz, V_{OUT} = 1V_{RMS}			–82			–82		dB
				0.008			0.008		%
Specifications for the LTC6912-2 ONLY									
Voltage Gain (Note 6)	V_S = 2.7V, Gain = 1, R_L = 10k	●	–0.07	0	0.07	–0.08	0	0.07	dB
	V_S = 2.7V, Gain = 1, R_L = 500Ω	●	–0.11	–0.02	0.07	–0.13	–0.02	0.07	dB
	V_S = 2.7V, Gain = 2, R_L = 10k	●	5.94	6.01	6.08	5.93	6.01	6.08	dB
	V_S = 2.7V, Gain = 4, R_L = 10k	●	11.9	12.02	12.12	11.88	12.02	12.12	dB
	V_S = 2.7V, Gain = 8, R_L = 10k	●	17.8	18.0	18.15	17.75	18.0	18.15	dB
	V_S = 2.7V, Gain = 8, R_L = 500Ω	●	17.65	17.94	18.15	17.50	17.94	18.15	dB
	V_S = 2.7V, Gain = 16, R_L =10k	●	23.8	24.01	24.25	23.75	24.01	24.25	dB
	V_S = 2.7V, Gain = 32, R_L = 10k	●	29.7	30.0	30.2	29.65	30.0	30.2	dB
	V_S = 2.7V, Gain = 64, R_L = 10k	●	35.4	35.8	36.2	35.15	35.8	36.2	dB
	V_S = 2.7V, Gain = 64, R_L = 500Ω	●	34.15	35.3	36.0	33.40	35.3	36.0	dB
	V_S = 5V, Gain = 1, R_L = 10k	●	–0.08	0	0.08	–0.09	0	0.08	dB
	V_S = 5V, Gain = 1, R_L = 500Ω	●	–0.1	–0.01	0.08	–0.12	–0.01	0.08	dB
	V_S = 5V, Gain = 2, R_L = 10k	●	5.95	6.02	6.09	5.94	6.02	6.09	dB
	V_S = 5V, Gain = 4, R_L = 10k	●	11.85	12.02	12.15	11.83	12.02	12.15	dB
	V_S = 5V, Gain = 8, R_L = 10k	●	17.85	18.01	18.15	17.83	18.01	18.15	dB
	V_S = 5V, Gain = 8, R_L = 500Ω	●	17.65	17.96	18.15	17.50	17.96	18.15	dB
	V_S = 5V, Gain = 16, R_L = 10k	●	23.85	24.02	24.15	23.80	24.02	24.15	dB
	V_S = 5V, Gain = 32, R_L = 10k	●	29.70	30.02	30.2	29.65	30.02	30.2	dB
	V_S = 5V, Gain = 64, R_L = 10k	●	35.5	35.9	36.25	35.40	35.9	36.25	dB
	V_S = 5V, Gain = 64, R_L = 500Ω	●	34.6	35.6	36.0	33.8	35.6	36.0	dB
	V_S = ±5V, Gain = 1, R_L = 10k	●	–0.06	0.01	0.08	–0.07	0.01	0.08	dB
	V_S = ±5V, Gain = 1, R_L = 500Ω	●	–0.1	0	0.08	–0.11	0	0.08	dB
	V_S = ±5V, Gain = 2, R_L = 10k	●	5.95	6.02	6.09	5.94	6.02	6.09	dB
	V_S = ±5V, Gain = 4, R_L = 10k	●	11.9	12.03	12.15	11.88	12.03	12.15	dB
	V_S = ±5V, Gain = 8, R_L = 10k	●	17.85	18.02	18.15	17.83	18.02	18.15	dB
	V_S = ±5V, Gain = 8, R_L = 500Ω	●	17.80	17.99	18.15	17.73	17.99	18.15	dB
	V_S = ±5V, Gain = 16, R_L = 10k	●	23.85	24.03	24.15	23.82	24.03	24.15	dB
	V_S = ±5V, Gain = 32, R_L = 10k	●	29.85	30.0	30.2	29.8	30.0	30.20	dB
	V_S = ±5V, Gain = 64, R_L = 10k	●	35.65	36.0	36.20	35.55	36.0	36.20	dB
	V_S = ±5V, Gain = 64, R_L = 500Ω	●	35.15	35.8	36.10	34.45	35.8	36.10	dB

ELECTRICAL CHARACTERISTICS

The ● denotes the specifications that apply over the full operating temperature range, otherwise specifications are at T_A = 25°C. V_S = 5V, AGND = 2.5V, Gain = 1, R_L = 10k to midsupply point, unless otherwise noted.

PARAMETER	CONDITIONS		C, I GRADES MIN	C, I GRADES TYP	C, I GRADES MAX	H GRADE MIN	H GRADE TYP	H GRADE MAX	UNITS
Specifications for the LTC6912-2 ONLY									
Channel-to-Channel Voltage Gain Match (Note 6)	V_S = 2.7V, Gain = 1, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 2.7V, Gain = 1, R_L = 500Ω	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 2.7V, Gain = 2, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 2.7V, Gain = 4, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 2.7V, Gain = 8, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 2.7V, Gain = 8, R_L = 500Ω	●	–0.15	±0.02	0.15	–0.2	±0.02	0.2	dB
	V_S = 2.7V, Gain = 16, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 2.7V, Gain = 32, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 2.7V, Gain = 64, R_L = 10k	●	–0.2	±0.02	0.2	–0.2	±0.02	0.2	dB
	V_S = 2.7V, Gain = 64, R_L = 500Ω	●	–0.7	±0.02	0.7	–1.0	±0.02	1.0	dB
	V_S = 5V, Gain = 1, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 5V, Gain = 1, R_L = 500Ω	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 5V, Gain = 2, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = 5V, Gain = 4, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 5V, Gain = 8, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 5V, Gain = 8, R_L = 500Ω	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 5V, Gain = 16, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 5V, Gain = 32, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 5V, Gain = 64, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = 5V, Gain = 64, R_L = 500Ω	●	–0.6	±0.02	0.6	–0.8	±0.02	0.8	dB
	V_S = ±5V, Gain = 1, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = ±5V, Gain = 1, R_L = 500Ω	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = ±5V, Gain = 2, R_L = 10k	●	–0.1	±0.02	0.1	–0.1	±0.02	0.1	dB
	V_S = ±5V, Gain = 4, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = ±5V, Gain = 8, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = ±5V, Gain = 8, R_L = 500Ω	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = ±5V, Gain = 16, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = ±5V, Gain = 32, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = ±5V, Gain = 64, R_L = 10k	●	–0.15	±0.02	0.15	–0.15	±0.02	0.15	dB
	V_S = ±5V, Gain = 64, R_L = 500Ω	●	–0.4	±0.02	0.4	–0.6	±0.02	0.6	dB
Gain Temperature Coefficient (Note 6)	V_S = 5V, Gain = 1, R_L = OPEN			2			2		ppm/°C
	V_S = 5V, Gain = 2, R_L = OPEN			–4			–4		ppm/°C
	V_S = 5V, Gain = 4, R_L = OPEN			–10			–10		ppm/°C
	V_S = 5V, Gain = 8, R_L = OPEN			–24			–24		ppm/°C
	V_S = 5V, Gain = 16, R_L = OPEN			–30			–30		ppm/°C
	V_S = 5V, Gain = 32, R_L = OPEN			–40			–40		ppm/°C
	V_S = 5V, Gain = 64, R_L = OPEN			–120			–120		ppm/°C
Channel-to-Channel Gain Temperature Coefficient Match (Note 6)	V_S = 5V, Gain = 1, R_L = OPEN			0			0		ppm/°C
	V_S = 5V, Gain = 2, R_L = OPEN			–0.5			–0.5		ppm/°C
	V_S = 5V, Gain = 4, R_L = OPEN			0			0		ppm/°C
	V_S = 5V, Gain = 8, R_L = OPEN			0			0		ppm/°C
	V_S = 5V, Gain = 16, R_L = OPEN			–1			–1		ppm/°C
	V_S = 5V, Gain = 32, R_L = OPEN			–4			–4		ppm/°C
	V_S = 5V, Gain = 64, R_L = OPEN			–4			–4		ppm/°C
Channel-to-Channel Isolation (Note 7)	f = 200kHz,								
	V_S = 5V, Gain = 1, R_L = 10k			117			117		dB
	V_S = 5V, Gain = 8, R_L = 10k			110			110		dB
	V_S = 5V, Gain = 64, R_L = 10k			92			92		dB
Offset Voltage Magnitude (Internal Op-Amp, Note 8)	Gain = 1	●		0.125	2		0.125	3.5	mV

ELECTRICAL CHARACTERISTICS

The ● denotes the specifications that apply over the full operating temperature range, otherwise specifications are at T_A = 25°C. V_S = 5V, AGND = 2.5V, Gain = 1, R_L = 10k to midsupply point, unless otherwise noted.

PARAMETER	CONDITIONS		C, I GRADES MIN	C, I GRADES TYP	C, I GRADES MAX	H GRADE MIN	H GRADE TYP	H GRADE MAX	UNITS
Specifications for the LTC6912-2 ONLY									
Offset Voltage Magnitude Referred to INA or INB Pins (Note 8)	Gain = 1	●		0.25	3.5		0.25	6.5	mV
	Gain = 8	●		0.14	2		0.14	4	mV
Input Offset Voltage Drift, Internal Op Amp				6			10		µV/°C
DC Input Resistance at INA or INB Pins (Note 9)	DC V_{INA} or V_{INB} = 0V								
	Gain = 0	●		>10			>10		MΩ
	State = 8, Software Shutdown	●		>10			>10		MΩ
	Gain = 1	●		10			10		kΩ
	Gain = 2	●		5			5		kΩ
	Gain = 4	●		2.5			2.5		kΩ
	Gain > 4	●		1.25			1.25		kΩ
DC Input Resistance Drift at INA or INB Pins (Note 9)	Gain = 1			85			95		ppm/°C
	Gain = 2			90			100		ppm/°C
	Gain = 4			95			105		ppm/°C
	Gain = 8			120			130		ppm/°C
	Gain = 16			130			140		ppm/°C
	Gain = 32			140			150		ppm/°C
	Gain = 64			170			180		ppm/°C
DC Input Resistance Match R_{INA}-R_{INB}	Gain = 1	●		10			10		Ω
	Gain = 2	●		5			5		Ω
	Gain = 4	●		5			5		Ω
	Gain > 4	●		5			5		Ω
DC Small Signal Output Resistance at OUT A or OUT B Pins	DC V_{INA} or V_{INB} = 0V								
	Gain = 0			0.4			0.4		Ω
	Gain = 1			0.7			0.7		Ω
	Gain = 2			1.0			1.0		Ω
	Gain = 4			1.9			1.9		Ω
	Gain = 8			3.4			3.4		Ω
	Gain = 16			6.4			6.4		Ω
	Gain = 32			15			15		Ω
	Gain = 64			30			30		Ω
	State = 8, Software Shutdown	●		>1			>1		MΩ
Gain Bandwidth Product	Gain = 64	●	17	30	50	15	30	50	MHz
Wideband Noise (Referred to Input)	f = 1kHz to 200kHz								
	Gain = 0 (Output Noise Only)			8.1			8.1		$µV_{RMS}$
	Gain = 1			13.8			13.8		$µV_{RMS}$
	Gain = 2			9.6			9.6		$µV_{RMS}$
	Gain = 4			7.5			7.5		$µV_{RMS}$
	Gain = 8			6.4			6.4		$µV_{RMS}$
	Gain = 16			6.0			6.0		$µV_{RMS}$
	Gain = 32			5.8			5.8		$µV_{RMS}$
	Gain = 64			5.6			5.6		$µV_{RMS}$

ELECTRICAL CHARACTERISTICS

The ● denotes the specifications that apply over the full operating temperature range, otherwise specifications are at T_A = 25°C. V_S = 5V, AGND = 2.5V, Gain = 1, R_L = 10k to midsupply point, unless otherwise noted.

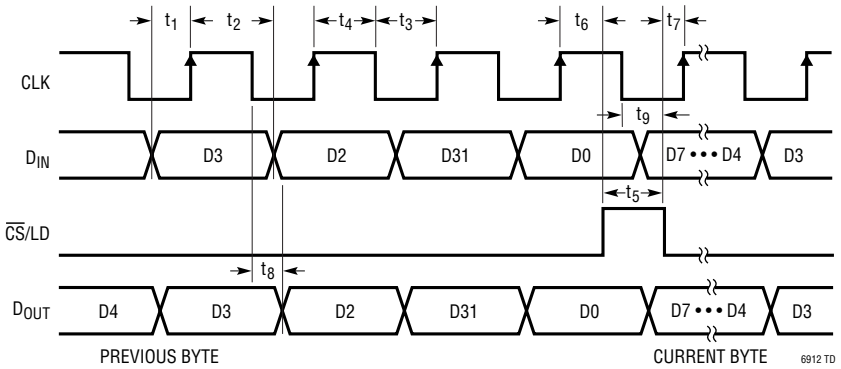
PARAMETER	CONDITIONS	C, I GRADES MIN	C, I GRADES TYP	C, I GRADES MAX	H GRADE MIN	H GRADE TYP	H GRADE MAX	UNITS
Specifications for the LTC6912-2 ONLY								
Voltage Noise Density (Referred to Input)	f = 50kHz							
	Gain = 1		31.1			31.1		nV/√Hz
	Gain = 2		22.8			22.8		nV/√Hz
	Gain = 4		17			17		nV/√Hz
	Gain = 8		14.6			14.6		nV/√Hz
	Gain = 16		13.2			13.2		nV/√Hz
	Gain = 32		12.9			12.9		nV/√Hz
	Gain = 64		12.6			12.6		nV/√Hz
Total Harmonic Distortion	Gain = 8, f_{IN} = 10kHz, V_{OUT} = 1V_{RMS}		–84 0.006			–84 0.006		dB %
	Gain = 8, f_{IN} = 100kHz, V_{OUT} = 1V_{RMS}		–82 0.008			–82 0.008		dB %

SERIAL INTERFACE SPECIFICATIONS

SYMBOL	PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
Digital I/O Logic Levels, All Digital I/O Voltage Referenced to DGND							
V_{IH}	Digital Input High Voltage		●	2			V
V_{IL}	Digital Input Low Voltage		●			0.8	V
V_{OH}	Digital Output High Voltage	Sourcing 500µA	●	V^+ – 0.3			V
V_{OL}	Digital Output Low Voltage	Sinking 500µA	●			0.3	V
Serial Interface Timing, V^+ = 2.7V ~ 4.5V, V^- = 0V (Note 10)							
t_1	D_{IN} Valid to CLK Setup		●	60			ns
t_2	D_{IN} Valid to CLK Hold		●	0			ns
t_3	CLK Low		●	100			ns
t_4	CLK High		●	100			ns
t_5	$\overline{CS}$/LD Pulse Width		●	60			ns
t_6	LSB CLK to $\overline{CS}$/LD		●	60			ns
t_7	$\overline{CS}$/LD Low to CLK		●	30			ns
t_8	D_{OUT} Output Delay	C_L = 15pF	●			125	ns
t_9	CLK Low to $\overline{CS}$/LD Low		●	0			ns
Serial Interface Timing, V^+ = 4.5V ~ 5.5V, V^- = 0V (Note 10)							
t_1	D_{IN} Valid to CLK Setup		●	30			ns
t_2	D_{IN} Valid to CLK Hold		●	0			ns
t_3	CLK Low		●	50			ns
t_4	CLK High		●	50			ns
t_5	$\overline{CS}$/LD Pulse Width		●	40			ns
t_6	LSB CLK to $\overline{CS}$/LD		●	40			ns
t_7	$\overline{CS}$/LD Low to CLK		●	20			ns
t_8	D_{OUT} Output Delay	C_L = 15pF	●			85	ns
t_9	CLK Low to $\overline{CS}$/LD Low		●	0			ns

SERIAL INTERFACE SPECIFICATIONS

SYMBOL	PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
Serial Interface Timing, Dual ±4.5V ~ ±5.5V Supplies (Note 10)							
t_1	D_{IN} Valid to CLK Setup		●	30			ns
t_2	D_{IN} Valid to CLK Hold		●	0			ns
t_3	CLK High		●	50			ns
t_4	CLK Low		●	50			ns
t_5	$\overline{CS}$/LD Pulse Width		●	40			ns
t_6	LSB CLK to $\overline{CS}$/LD		●	40			ns
t_7	$\overline{CS}$/LD Low to CLK		●	20			ns
t_8	D_{OUT} Output Delay	C_L = 15pF	●			85	ns
t_9	CLK Low to $\overline{CS}$/LD Low		●	0			ns



Note 1: Absolute Maximum Ratings are those values beyond which the life of the device may be impaired.

Note 2: The LTC6912-1C and LTC6912-1I are guaranteed functional over the operating temperature range of –40°C to 85°C. The LTC6912-1H is guaranteed functional over the operating temperature range of –40°C to 125°C.

Note 3: The LTC6912-1C is guaranteed to meet specified performance from 0°C to 70°C. The LTC6912-1C is designed, characterized and expected to meet specified performance from –40°C to 85°C but is not tested or QA sampled at these temperatures. The LTC6912-1I is guaranteed to meet specified performance from –40°C to 85°C. The LTC6912-1H is guaranteed to meet specified performance from –40°C to 125°C.

Note 4: Output voltage swings are measured as differences between the output and the respective supply rail.

Note 5: Extended operation with output shorted may cause junction temperature to exceed the 150°C limit for GN package and 125°C for a DFN package is not recommended.

Note 6: Gain is measured with a large signal DC test using an output excursion between approximately 30% and 70% of supply voltage.

Note 7: Channel-to-channel isolation is measured by applying a 200kHz input signal to one channel so that its output varies $1V_{RMS}$, and measuring the output voltage RMS of the other channel relative to AGND with its input tied to AGND. Isolation is calculated:

$Isolation_B = 20 \bullet \log^{10}(V_{OUTA}/V_{OUTB})$ or
$Isolation_A = 20 \bullet \log^{10}(V_{OUTB}/V_{OUTA})$

High channel-to-channel isolation is strongly dependent on proper circuit layout. See Applications Information.

Note 8: Offset voltage referred to the INA or INB input is (1 + 1/|GAIN|) times the offset voltage of the internal op amp, where GAIN is the nominal gain magnitude. The typical offset voltage values are for 25°C only. See Applications Information.

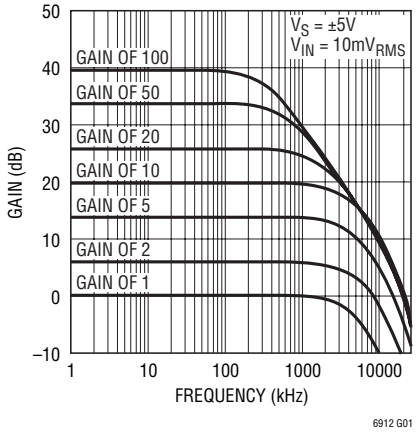
Note 9: Input resistance can vary by approximately ±30% part-to-part at a given gain setting.

Note 10: Guaranteed by design, not subject to test.

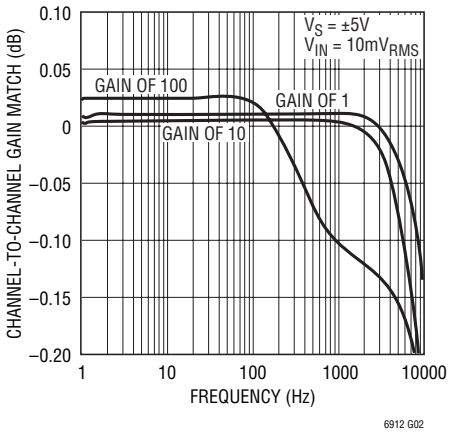
Note 11: States 13, 14 and 15 (binary 11xx) are not used. Programming a channel to states 8 or higher will configure that particular channel into a low power shutdown state. In addition, programming a channel into state 15 (binary 1111) will cause that particular channel to draw up to 20mA of supply current and is not recommended.

TYPICAL PERFORMANCE CHARACTERISTICS

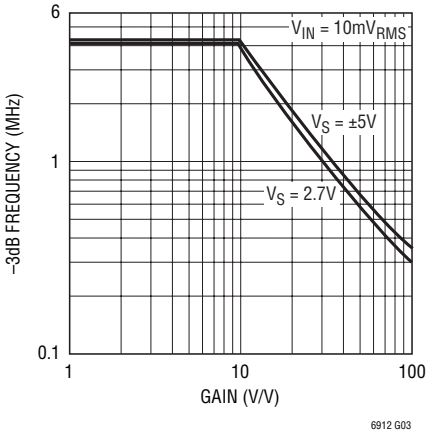
LTC6912-1 Frequency Response



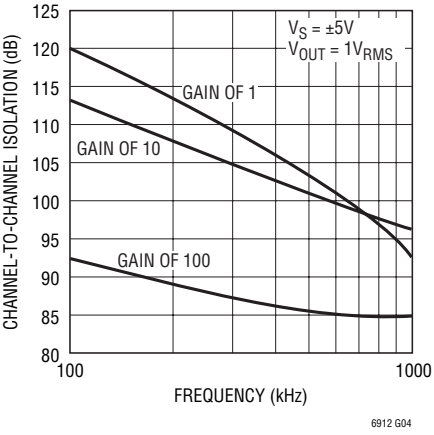
LTC6912-1 Channel Gain Matching vs Frequency



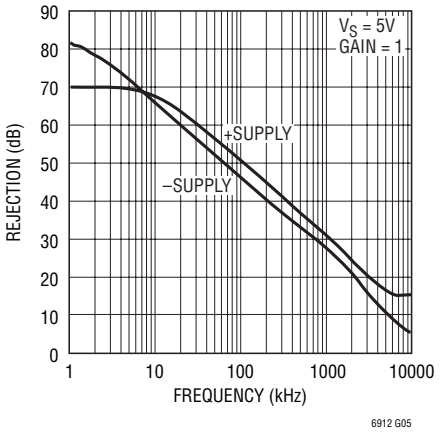
LTC6912-1 –3dB Bandwidth vs Gain Setting



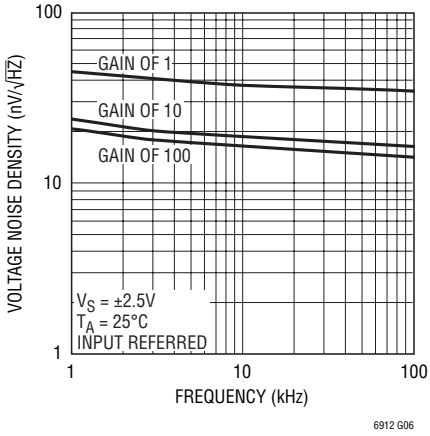
LTC6912-1 Channel Isolation vs Frequency



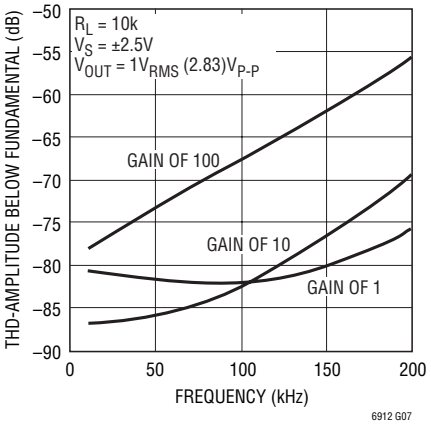
LTC6912-1 Power Supply Rejection vs Frequency



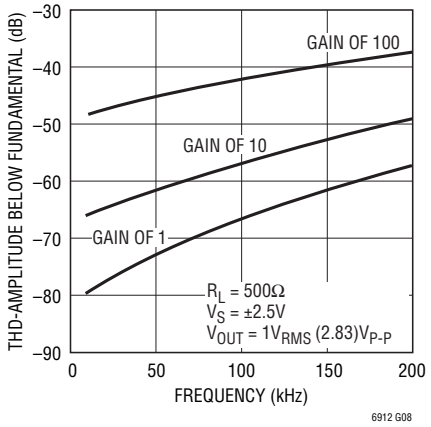
LTC6912-1 Noise Density vs Frequency



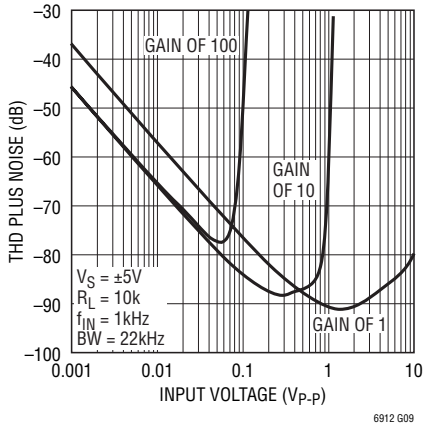
LTC6912-1 Distortion vs Frequency with Light Loading



LTC6912-1 Distortion vs Frequency with Heavy Loading

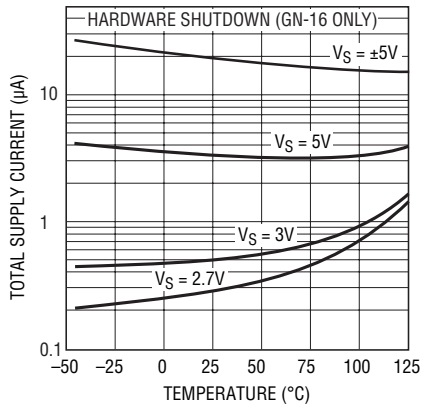


LTC6912-1 THD Plus Noise vs Input Voltage

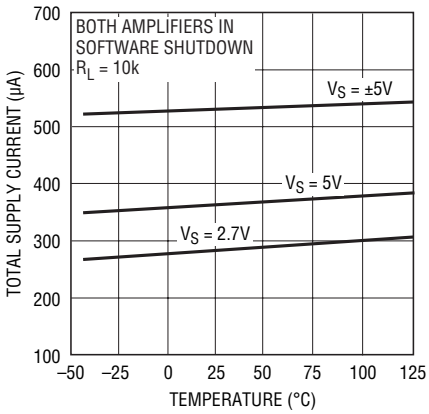


TYPICAL PERFORMANCE CHARACTERISTICS

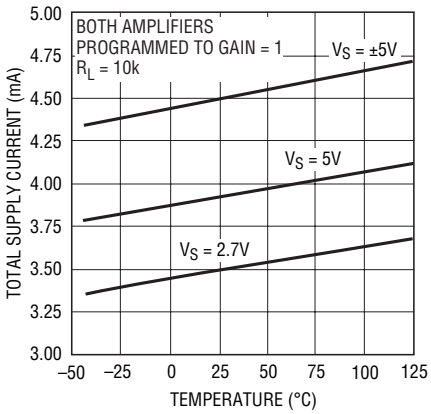
LTC6912-1 Hardware Shutdown Total Supply Current vs Temperature



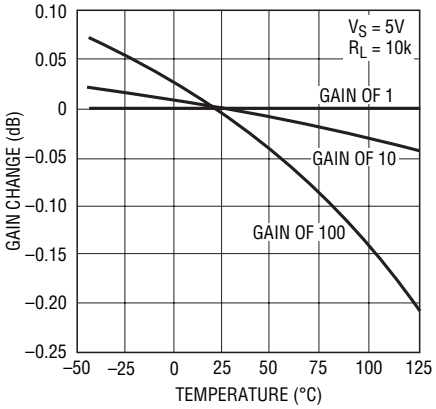
LTC6912-1 Software Shutdown Total Supply Current vs Temperature



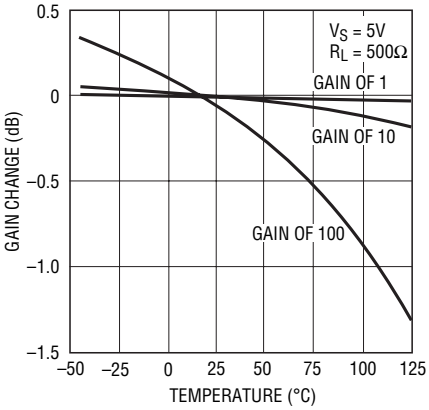
LTC6912-1 Total Supply Current vs Temperature (Both Amplifiers Active)



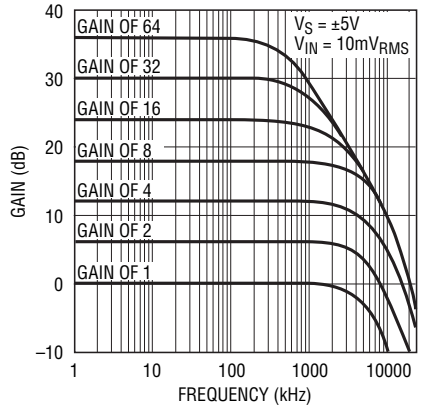
LTC6912-1 Gain Shift vs Temperature (Light Load)



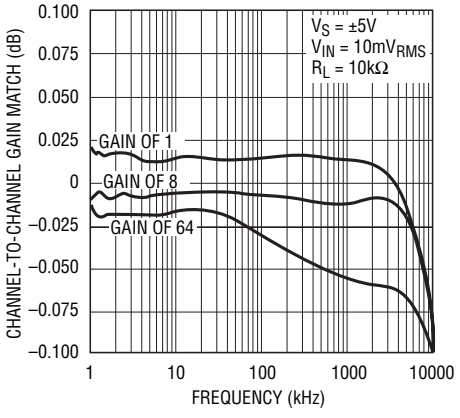
LTC6912-1 Gain Shift vs Temperature (Heavy Load)



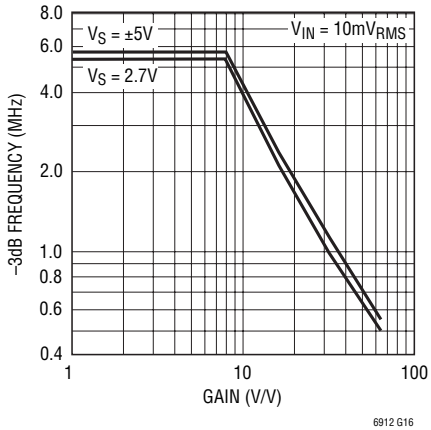
LTC6912-2 Frequency Response



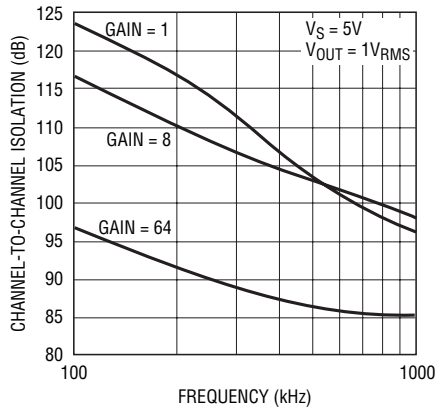
LTC6912-2 Channel Gain Matching vs Frequency



LTC6912-2 –3dB Bandwidth vs Gain Setting

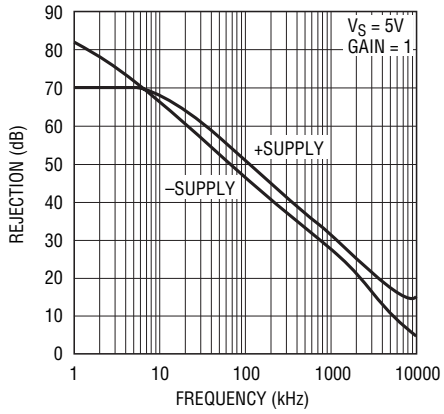


LTC6912-2 Channel Isolation vs Frequency

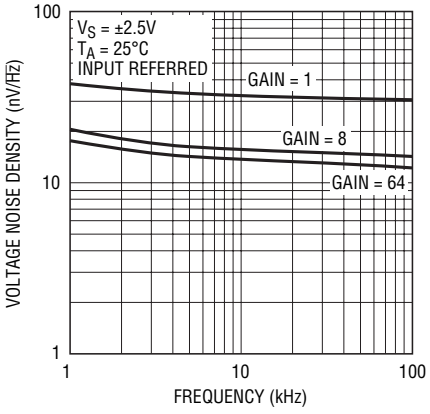


TYPICAL PERFORMANCE CHARACTERISTICS

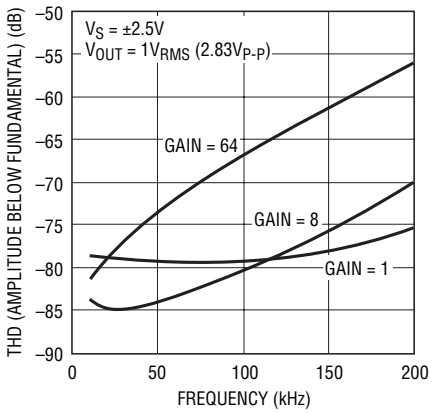
LTC6912-2 Power Supply Rejection vs Frequency



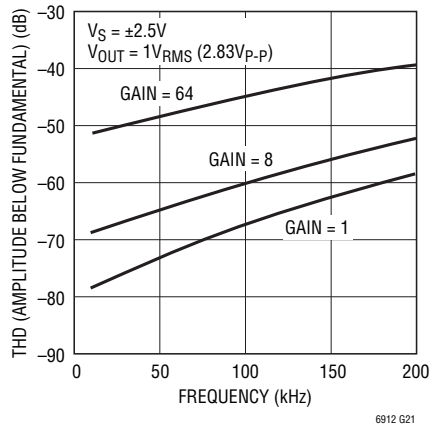
LTC6912-2 Noise Density vs Frequency



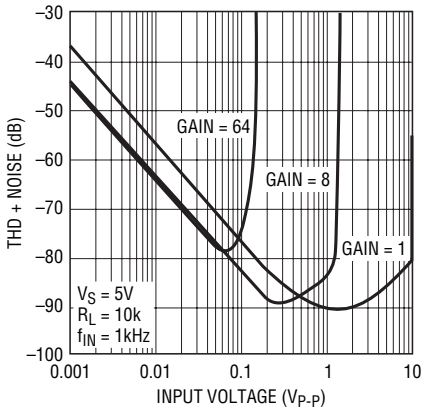
LTC6912-2 Distortion vs Frequency with Light Loading (R_L = 10k)



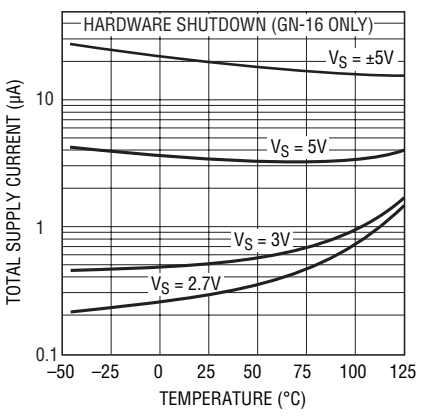
LTC6912-2 Distortion vs Frequency with Heavy Loading (R_L = 500Ω)



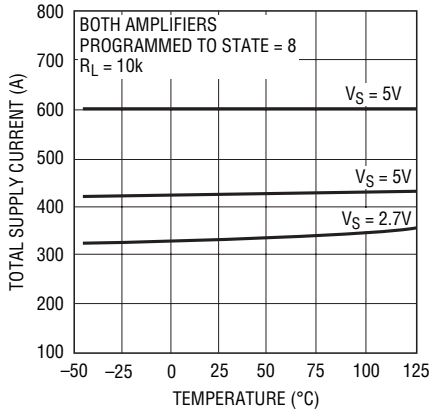
LTC6912-2 THD + Noise vs Input Voltage



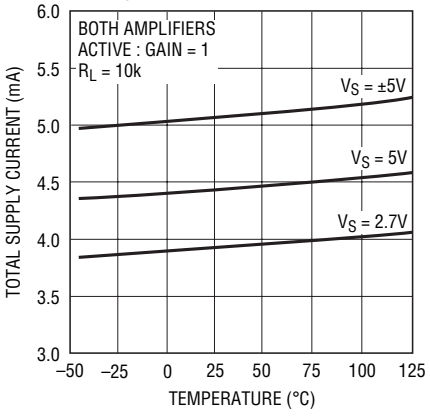
LTC6912-2 Hardware Shutdown Total Supply Current vs Temperature



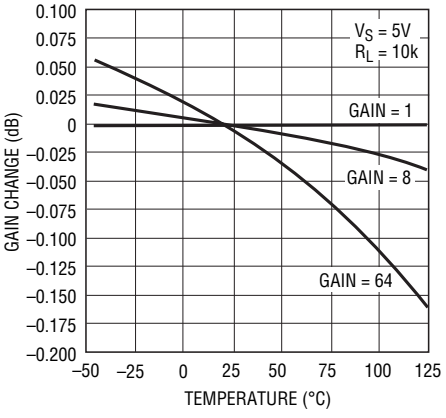
LTC6912-2 Software Shutdown Total Supply Current vs Temperature



LTC6912-2 Total Supply Current vs Temperature (Both Amplifiers Active)

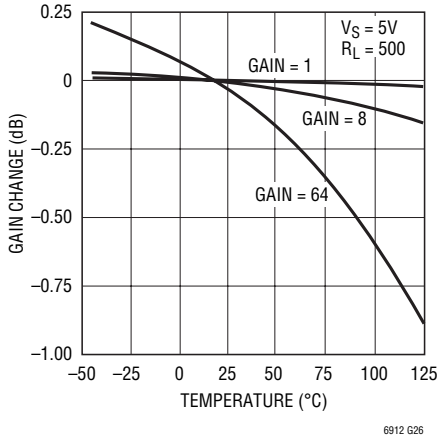


LTC6912-2 Gain Shift vs Temperature (Light Load)



TYPICAL PERFORMANCE CHARACTERISTICS

LTC6912-2 Gain Shift vs Temperature (Heavy Load)



PIN FUNCTIONS

INA, INB: Analog Inputs. The input signal to the A channel amplifier of the LTC6912-X is the voltage difference between the INA pin and AGND pin. Likewise, the input signal to the B channel amplifier of the LTC6912-X is the voltage difference between the INB pin and AGND pin. The INA (or INB) pin connects internally to a digitally controlled resistance whose other end is a current summing point at the same potential as the AGND pin (Figure 1). At unity gain, the value of this input resistance is approximately 10kΩ and the INA (or INB) pin voltage range is rail-to-rail (V^+ to V^-). At gain settings above unity, the input resistance falls. The linear input range at INA and INB also falls inversely proportional to the programmed gain. Tables 1 and 2 summarize this behavior. The higher gains are designed to boost lower level signals with good noise performance. In the "zero" gain state (state = 0), or in software shutdown (state = 8) analog switches disconnect the INA or INB pin internally and this pin presents a very high input resistance. In the "zero" gain state (state = 0), the input may vary from rail to rail but the output is insensitive to it and is forced to the AGND potential. Circuitry driving the INA and INB pins must consider the LTC6912-X's input resistance, its process variance, and the variation of this resistance from gain setting to gain setting. Signal sources with significant output resistance may introduce a gain error as the source's output resistance and the LTC6912-X's input resistance forms a voltage divider. This is especially true at higher gain settings where the input resistance is the lowest.

In single supply voltage applications, the LTC6912-X's DC ground reference for both input and output is AGND, not V^-. With increasing gains, the LTC6912-X's input voltage range for an unclipped output is no longer rail-to-rail but diminishes inversely to gain, centered about the AGND potential.

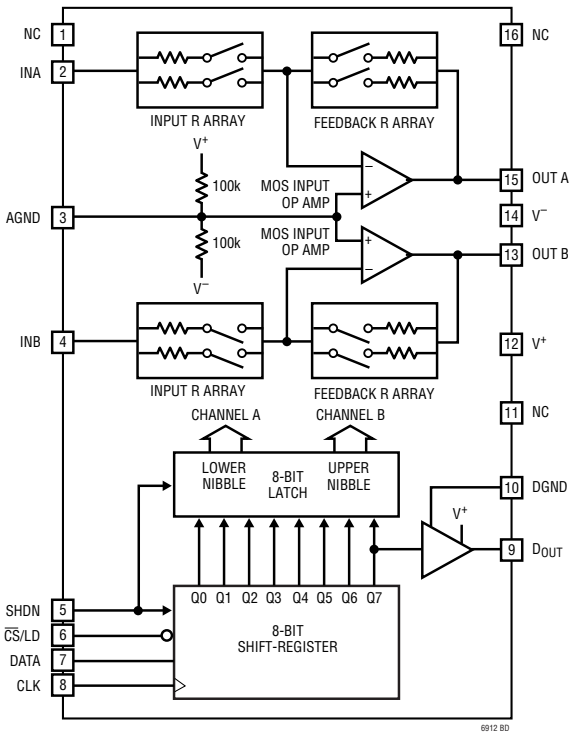


Figure 1. GN-16 Block Diagram

PIN FUNCTIONS

AGND: Analog Ground. The AGND pin is at the midpoint of an internal resistive voltage divider, developing a potential halfway between the V^+ and V^- pins. In normal operation, the AGND pin has an equivalent input resistance of nominally 50k (Figure 1). In order to reduce the quiescent supply current in hardware shutdown (SHDN pin pulled to V^+, GN-16 only), the equivalent series resistance of this pin significantly increases (to a value on the order of 800kΩ with 5V supplies, but is highly supply voltage, temperature, and process dependent). AGND is the noninverting input to both the internal channel A and channel B amplifiers. This makes AGND the ground reference voltage for the INA, INB, OUTA, and OUTB pins. Recommended analog ground plane connection depends on how power is applied to the LTC6912-X (See Figures 2, 3, and 4). Single power supply applications typically use V^- for the system signal ground. The analog ground plane in single-supply applications should therefore tie to V^-, and the AGND pin should be bypassed to this ground plane by a high quality capacitor of at least 0.1µF (Figure 2). The AGND pin provides an internal analog reference voltage at half the V^+ supply voltage. Dual supply applications with symmetrical supplies (such as ±5V) have a natural system ground plane potential of zero volts, in which the AGND pin can be directly tied to, making the zero volt ground plane the input and output reference voltage for the LTC6912-X (Figure 3). Finally, if dual asymmetrical power supplies are used, the supply ground is still the natural ground plane voltage. To maximize signal swing capability with an asymmetrical supply, however, it is often desirable to refer the LTC6912-X's analog input and output to a voltage equidistant from the two supply rails V^+ and V^-. The AGND pin will provide such a potential when open-circuited and bypassed with a capacitor (Figure 4). In noise sensitive applications where AGND does not tie directly to a ground plane, as in Figures 2 and 4, it is important to AC-bypass the AGND pin. Otherwise channel to channel isolation is degraded, and wideband noise will enter the signal path from the thermal noise of the internal voltage divider resistors which present a Thévenin equivalent resistance of approximately 50kΩ. This noise can reduce SNR by at least 15dB at high gain settings. An external capacitor from AGND to the ground plane, whose impedance is well below 50kΩ at frequencies of interest, will filter and suppress this noise. A 0.1µF high quality capacitor is effective for frequencies down to 1kHz. Larger capacitors will extend this suppression to lower frequencies. This issue does not arise in dual supply applications because the AGND pin ties directly to ground. In applications requiring an analog ground reference other than half the total supply voltage, the user can override the built-in analog ground reference by tying the AGND pin to a reference voltage with the AGND voltage range specified in the Electrical Characteristics Table. The AGND pin will load the external reference with approximately 50kΩ returned to the half-supply potential. AGND should still be capacitively bypassed to a ground plane as noted above. Do not connect the AGND pin to the V^- pin.

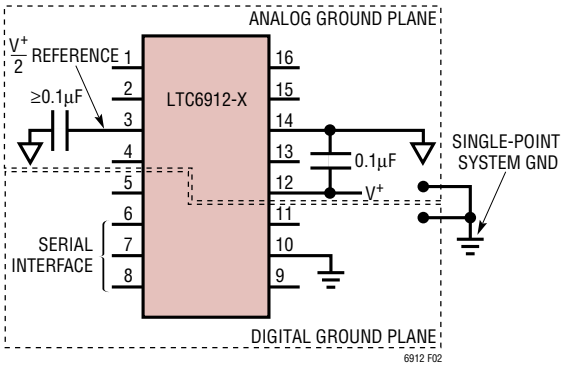


Figure 2. Single Supply Ground Plane Connection

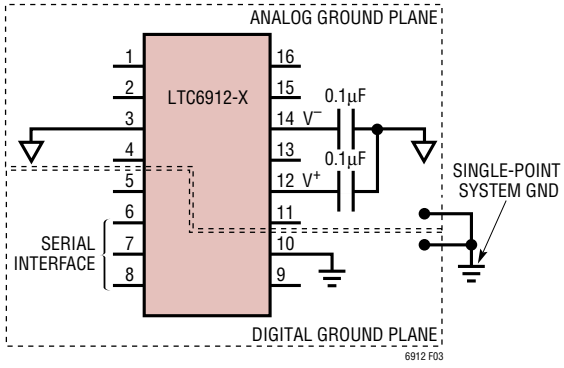


Figure 3. Symmetrical Dual Supply Ground Plane Connection

PIN FUNCTIONS

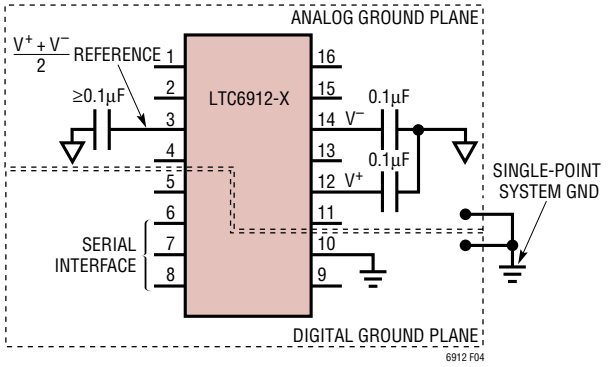


Figure 4. Asymmetrical Dual Supply Ground Plane Connection

SHDN (GN-16 ONLY): CMOS Compatible Logic Hardware Shutdown Input. The LTC6912-X has two shutdown modes. One is a software shutdown state which can be software programmed into either Channel A, Channel B, or both. The software shutdown, when programmed to a particular channel (state = 8), will disable that channel's amplifier and tri-state open its analog input and analog output. The serial interface, however is still active. A hardware shutdown occurs when the SHDN pin is pulled to the positive rail. In this condition, both amplifiers and serial interface are disabled. The SHDN pin is allowed to swing from V^- to 10.5V above V^-, regardless of V^+ so long as the logic levels meet the minimum requirements specified in the Electrical Characteristics table. The SHDN pin is a high impedance CMOS logic input, but has a small pull-down current source (<10µA) which will force SHDN low if the logic input is externally floated. On initial power up (with SHDN open), or coming out of the hardware shutdown mode (pulling SHDN to V^-), both amplifiers are reset into the power-on reset state (software shutdown mode, state = 8) for both channels.

$\overline{CS}$/LD: TTL/CMOS Compatible Logic Input. When this pin is asserted low, the CLK pin is enabled, and the 8-bit shift register serially shifts the shift register contents and whatever data is present on the D_{IN} pin into the shift register on the rising edge of CLK. On the rising edge of $\overline{CS}$/LD, the contents of the shift register data are loaded into the eight bit latch which configures the gain state of both channel A and channel B amplifiers. A logic high on $\overline{CS}$/LD inhibits the CLK signal internally to the IC.

D_{IN}: TTL/CMOS Compatible Logic Serial Data Input. The serial interface is synchronously loaded MSB first via D_{IN} on the rising edge of CLK with $\overline{CS}$/LD asserted low.

CLK: TTL/CMOS Compatible Logic Input. With $\overline{CS}$/LD asserted low, the clock synchronizes the loading of the serial shift register on its rising and falling edges. Data is shifted in at D_{IN} on the rising edge of CLK and is shifted out on D_{OUT} on the falling edge of CLK.

D_{OUT}: TTL/CMOS Compatible Logic Output. The MSB of the shift register contents is shifted out at D_{OUT} on the falling edge of CLK. The output at D_{OUT} swings between V^+ and DGND, and is rated to drive approximately 15pF.

DGND: Digital Ground: The DGND pin defines the potential from which LOGIC levels V_{IH} and V_{IL} for the 3-wire serial digital interface are referenced. The recommended connection of DGND depends on how power is applied to the LTC6912 (See Figures 2, 3, and 4). (CAVEAT: Under no conditions is DGND to exceed either supply pins V^+ and V^-, which could result in damage to the IC if not current limited.)

Single power supply applications typically use V^- for the system signal ground. The preferred connection for DGND is therefore V^- (See Figure 2).

Dual supply applications with symmetrical supplies (such as ±5V) have a natural system ground potential of zero volts, in which the DGND pin can be tied to, making the zero volt ground plane the logic reference (Figure 3).

Finally, if dual asymmetrical power supplies are used, the system ground is still the natural ground plane voltage.

V^-, V^+: Power Supply Pins. The V^+ and V^- pins should be bypassed with 0.1µF capacitors to an adequate analog ground plane using the shortest possible wiring. Electrically clean supplies and a low impedance ground are important for the high dynamic range available from the LTC6912 (see further details under the AGND pin description). Low noise linear power supplies are recommended. Switching power supplies require special care to prevent switching noise coupling into the signal path, reducing dynamic range.

PIN FUNCTIONS

OUT A, OUT B: Analog Output. These pins are the output of the A and B channel amplifiers respectively. Each operational amplifier can swing rail-to-rail (V^+ to V^-) as specified in the Electrical Characteristics table. For best performance, loading the output as lightly as possible will minimize signal distortion and gain error. The Electrical Characteristics table shows performance at output currents up to 10mA, and the current limits which occur when the output is shorted midsupply at 2.7V and ±5V supplies. Output current above 10mA is possible but current-limiting circuitry will begin to affect amplifier performance at approximately 20mA. Long-term operation above 20mA output is not recommended. Do not exceed maximum junction temperature of 150°C for a GN and 125°C for a DFN package. The output will drive capacitive loads up to 50pF. Capacitances higher than 50pF should be isolated by a series resistor (10Ω or higher).

APPLICATIONS INFORMATION

Functional Description

The LTC6912-X is a small outline, wideband, inverting two-channel amplifier with voltage gains that are independently programmable. Each delivers a choice of eight voltage gains, configurable through a 3-wire serial digital interface, which accepts TTL or CMOS logic levels (See Figure 5). Tables 1 and 2 list the nominal gains for the LTC6912-1 and LTC6912-2 respectively. Gain control within the amplifier occurs by switching resistors from a matched array in or out of a closed-loop op amp circuit using MOS analog switches (Figure 1). The bandwidths of the individual amplifiers depend on gain setting. The Typical Performance Characteristics section shows measured frequency responses.

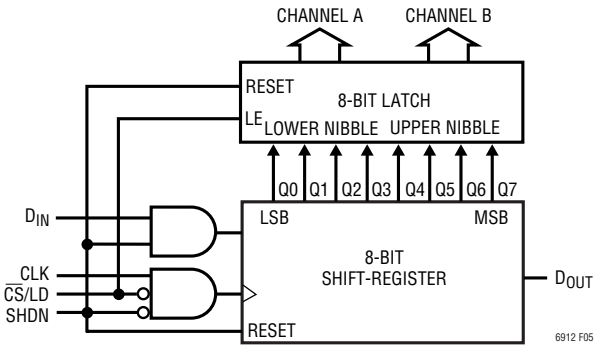


Figure 5. Serial Digital Interface Block Diagram

Description of the 3-Wire SPI Interface

Gain control of each amplifier is independently programmable using the 3-wire SPI interface (see Figure 5). Logic levels for the LTC6912 3-wire serial interface are TTL/CMOS compatible. When $\overline{CS}$/LD is low, the serial data on D_{IN} is shifted into an 8-bit shift-register on the rising edge of the clock, with the MSB transferred first. Serial data on D_{OUT} is shifted out on the clock's falling edge. A rising edge on $\overline{CS}$/LD will latch the shift-register's contents into an 8-bit D-latch and disable the clock internally on the IC. The upper nibble of the D-latch (4 most significant bits), configure the gain for the B-channel amplifier. The lower nibble of the D-latch (4 least significant bits), configures the gain for the A-channel amplifier. Tables 1 and 2 detail the nominal gains and respective gain codes. Care must be taken to ensure CLK is taken low before $\overline{CS}$/LD is pulled low to avoid an extra internal clock pulse to the input of the 8-bit shift-register (See Figure 5).

D_{OUT} is active in all states, therefore D_{OUT} cannot be "wire-OR'd" to other SPI outputs.

An LTC6912 may be daisy-chained with other LTC6912s or other devices having serial interfaces by connecting the D_{OUT} to the D_{IN} of the next chip while CLK and $\overline{CS}$/LD remain common to all chips in the daisy chain. The serial data is clocked to all the chips then the $\overline{CS}$/LD signal is pulled high to update all of them simultaneously. Figure 6 shows an example of two LTC6912s in a daisy chained SPI

APPLICATIONS INFORMATION

configuration. It is recommended the serial interface signals should remain idle in between data transfers in order to minimize digital noise coupling into the analog path.

Power On Reset

On the initial application of power, the power on reset state of both amplifiers is low power software shutdown (state = 8) (see Tables 1 and 2). In this state, both analog amplifiers are disabled and have their inputs and outputs opened. This will facilitate the application of using the device as a 2:1 analog MUX, in that the amplifier's outputs may be wired-OR together and the LTC6912 can alternately select between A and B channels. Care must be taken if the outputs are wired-OR'd to ensure the software shutdown state (state = 8) is always programmed in one of the two channels.

Timing Constraints

Settling time in the CMOS gain-control logic is typically several nanoseconds and is faster than the analog signal path. When the amplifier gain changes, the limiting timing is analog. As with any programmable-gain amplifier, each gain change causes an output transient as the amplifier's output moves, with finite speed, toward a differently scaled version of the input signal. The LTC6912-X analog path settles with a characteristic time constant or time scale, τ, that is roughly the standard value for a first order band limited response:

$$\tau = 0.35/f_{-3dB}$$

See the –3dB BW vs Gain Setting graph in the Typical Performance Characteristics section.

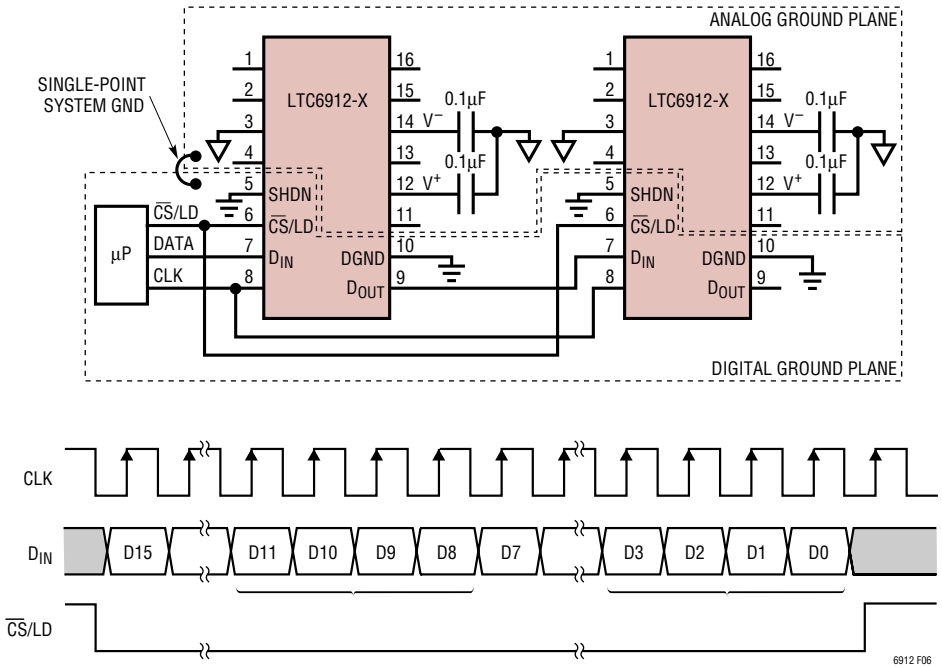


Figure 6. Two LTC6912s (Four PGAs) in Daisy Chain Configuration

APPLICATIONS INFORMATION

Offset Voltage vs Gain Setting

The electrical tables list DC offset (error), $V_{OS(OA)}$, at the inputs of the internal op amp (See Figure 1). The electrical tables also show the resulting, gain dependent offset voltage referred to the INA, or INB pins, $V_{OS(IN)}$. The two measures are related through the feedback/input resistor ratio, which equals the nominal gain-magnitude setting, |GAIN|:

$$V_{OS(IN)} = (1 + 1/|GAIN|)\ V_{OS(OA)}$$

Offset voltages at any gain setting can be inferred from this relationship. For example, an internal amplifier offset $V_{OS(OA)}$ of 1mV will appear referred to the INA, INB pins as 2mV at a gain setting of 1, or 1.5mV at a gain setting of 2. At high gains, $V_{OS(IN)}$ approaches $V_{OS(OA)}$. (Offset voltage is random and can have either polarity centered on 0V). The MOS input circuitry of the internal op amp in Figure 1 draws negligible input currents (less than 10µA), so only $V_{OS(OA)}$ and the GAIN affect the overall amplifier's offset.

AC-Coupled Operation

Adding capacitors in series with the INA and INB pins converts the LTC6912-X into a dual AC-coupled inverting amplifier, suppressing the input signal's DC level (and also adding the additional benefit of reducing the offset voltage from the LTC6912-X's amplifier itself). No further components are required because the input of the LTC6912-X biases itself correctly when a series capacitor is added. The INA and INB analog input pins connect internally to a resistor whose nominal value varies between 10kΩ and 1kΩ depending on the version of LTC6912 used (see the rightmost column of Tables 1 and 2). Therefore, the low frequency cutoff will vary with capacitor and gain setting. If, for example, a low frequency corner of 1kHz (or lower) on the LTC6912-1 is desired, use a series capacitor of 0.16µF or larger. 0.16µF has a reactance of 1kΩ at 1kHz, giving a 1kHz lower –3dB frequency for gain settings of 10V/V through 100V/V. If the LTC6912-1 is operated at lower gain settings with a 0.16µF capacitor, the higher input resistance will reduce the lower corner frequency down to 100Hz at a gain setting of 1V/V. These frequencies scale inversely with the value of input capacitor used.

Note that operating the LTC6912 family in "zero" gain mode (digital state 0000) open circuits both the INA and INB pins and this demands some care if employed with a series AC coupling input capacitor. When the chip enters the zero gain mode, the opened INA or INB pin tends to sample and freeze the voltage across the capacitor to the value it held just before the zero gain state. This can place the INA or INB pin at or near the DC potential of a supply rail. (The INA or INB pin may also drift to a supply potential in this state due to small leakage currents.) To prevent driving the INA or INB pin outside the supply limit and potentially damaging the chip, avoid AC input signals in the zero gain state with an AC coupling capacitor. Also, switching later to a non-zero gain value will cause a transient pulse at the output of the LTC6912-1 (with a time constant set by the capacitor value and the new LTC6912-1 input resistance value). This occurs because the INA and INB pins return to the AGND potential forcing transient current sourced by the amplifier output to charge the AC coupling capacitor to its proper DC blocking value.

SNR and Dynamic Range

The term "dynamic range" is much used (and abused) with signal paths. Signal-to-noise (SNR) is an unambiguous comparison of signal and noise levels, measured in the same way and under the same operating conditions. In a variable gain amplifier, however, further characterization is useful because both noise and maximum signal level in the amplifier will vary with the gain setting, in general. In the LTC6912-X, maximum output signal is independent of gain (and is near the full power supply voltage, as detailed in the swing sections of the Electrical Characteristics table). The maximum input level falls with increasing gain, and the input-referred noise falls as well (listed also in the table). To summarize the useful signal range in such an amplifier, we define dynamic range (DR) as the ratio of maximum input (at unity gain) to minimum input-referred noise (at maximum gain). This DR has a physical interpretation as the range of signal levels that will experience an SNR above unity V/V or 0dB. At a 10V total power supply, DR in the LTC6912-X (gains 0V/V to 100V/V), the DR is typically 115dB (the ratio of 9.9 V_{P-P}, or 3.5V_{RMS}, maximum input to the 6.3µV_{RMS} high gain input noise). The

APPLICATIONS INFORMATION

SNR from an amplifier is the ratio of input level to input-referred noise, and can be 108dB with the LTC6912 family at unity gain.

Construction and Instrumentation Cautions

Electrically clean construction is important in applications seeking the full dynamic range of the LTC6912 family of dual amplifiers. It is absolutely critical to have AGND either AC bypassed or wired directly using the shortest possible wiring, to a low impedance ground return for best channel-to-channel isolation. Short, direct wiring minimizes parasitic capacitance and inductance. High quality supply bypass capacitors of 0.1µF near the chip provide good decoupling from a clean, low inductance power source. But several centimeters of wire (i.e., a few µH of inductance) from the power supplies, unless decoupled by substantial capacitance (>10µF) near the chip, can create a parasitic high-Q LC resonant circuit in the hundreds of kHz range in the chip's supplies or ground reference. This may impair circuit performance at those frequencies. A compact, carefully laid out printed circuit board with a good ground plane makes a significant difference in minimizing distortion. Finally, equipment to measure performance can itself introduce distortion or noise floors. Checking for these limits with wired shorts from INA to OUTA and INB to OUTB in place of the chip is a prudent routine procedure.

TYPICAL APPLICATION

Low Noise AC Amplifier with Programmable Gain and Bandwidth

Analog data acquisition can exploit band limiting as well as gain to suppress unwanted signals or noise. Tailoring an analog front end to both the level and bandwidth of each source maximizes the resulting SNR. Figure 7 shows a block diagram for a low noise amplifier with gain and bandwidth independently programmable over a 100:1 range. Channels A and B of the LTC6912-1 are used to independently control the gain and bandwidth respectively over a 100:1 range. The LT1884 dual op amp forms an integrating lowpass loop with capacitor C2 to set the programmable upper corner frequency. The LT1884 also supports rail-to-rail output swings over the total supply voltage range of 2.7V to 10.5V. AC coupling through capacitor C1 establishes a fixed low frequency corner of 1Hz, which can be adjusted by changing C1. Alternatively, shorting C1 makes the amplifier DC coupled. If DC gain is not needed, the AC coupling cap C1 serves to suppress several error sources: any shift in DC levels, low frequency noise, and DC offset voltages (not including the LT1884's low internal offset).

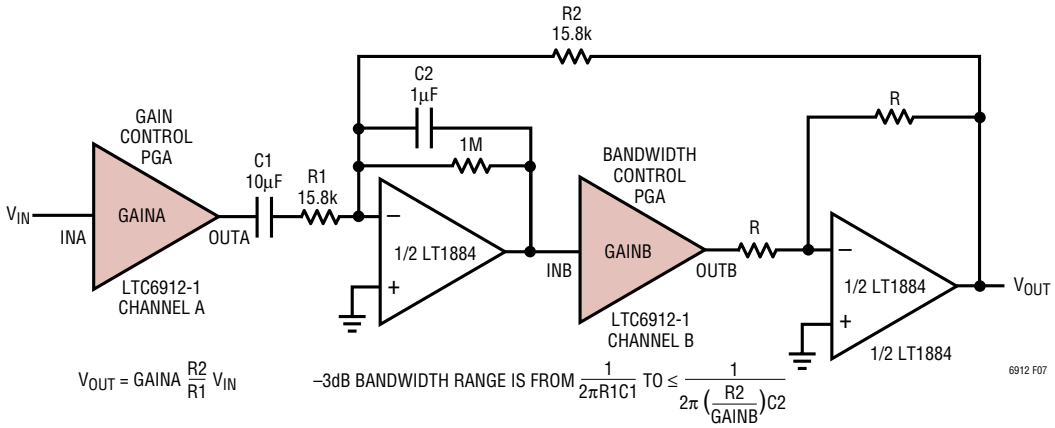
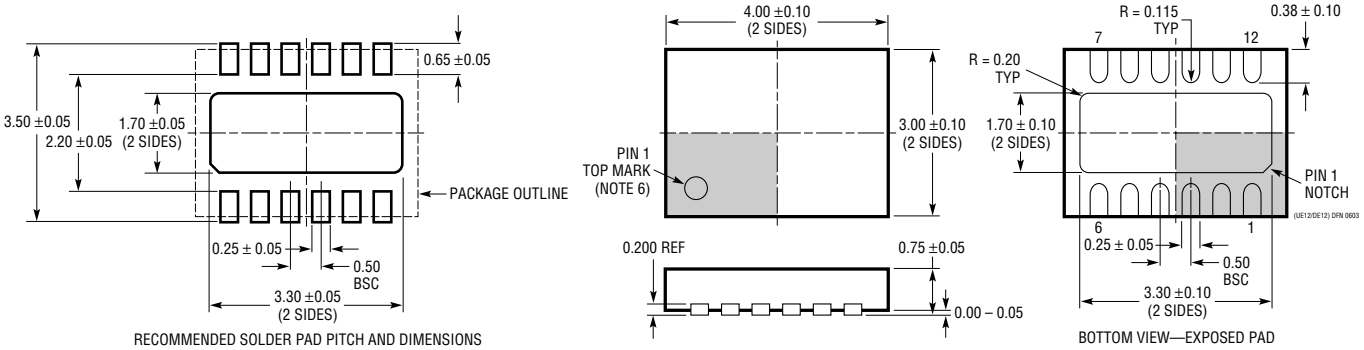


Figure 7. Block Diagram of an AC Amplifier with Programmable Gain and Bandwidth

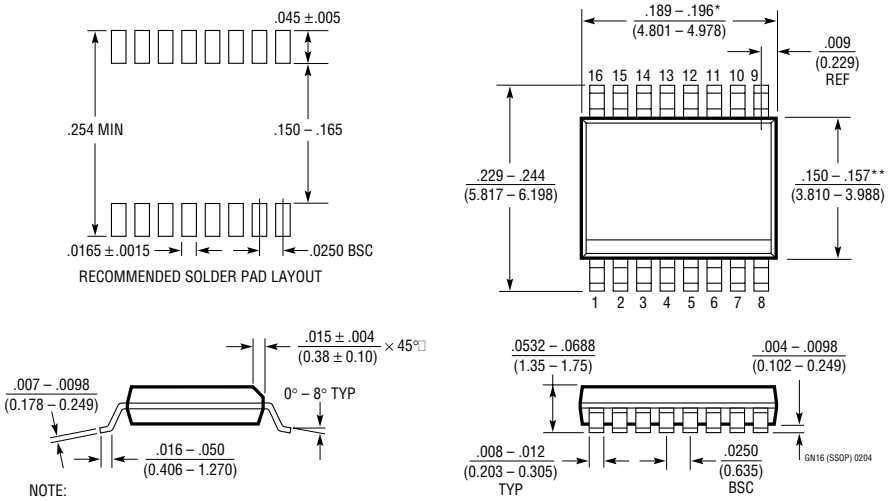
PACKAGE DESCRIPTION

DE/UE Package
12-Lead Plastic DFN (4mm × 3mm)
(Reference LTC DWG # 05-08-1695)



NOTE:
1. DRAWING PROPOSED TO BE A VARIATION OF VERSION (WGED) IN JEDEC PACKAGE OUTLINE M0-229
2. DRAWING NOT TO SCALE
3. ALL DIMENSIONS ARE IN MILLIMETERS
4. DIMENSIONS OF EXPOSED PAD ON BOTTOM OF PACKAGE DO NOT INCLUDE MOLD FLASH. MOLD FLASH, IF PRESENT, SHALL NOT EXCEED 0.15mm ON ANY SIDE
5. EXPOSED PAD SHALL BE SOLDER PLATED
6. SHADED AREA IS ONLY A REFERENCE FOR PIN 1 LOCATION ON THE TOP AND BOTTOM OF PACKAGE

GN Package
16-Lead Plastic SSOP (Narrow .150 Inch)
(Reference LTC DWG # 05-08-1641)



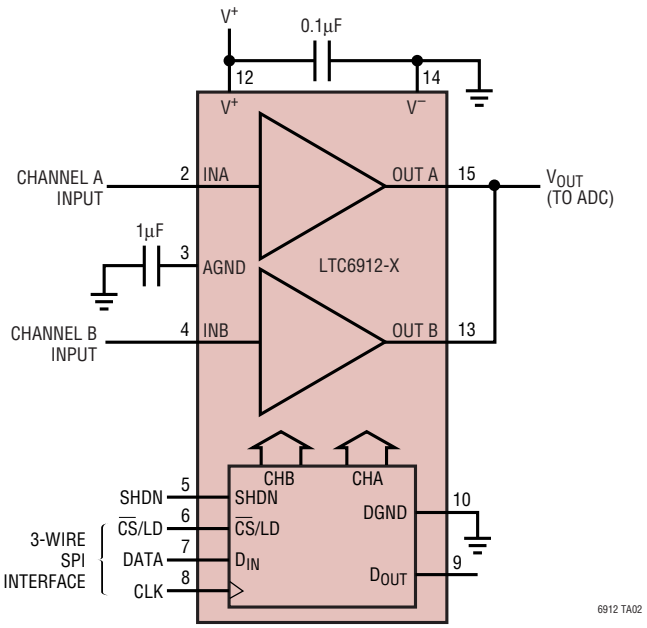
NOTE:
1. CONTROLLING DIMENSION: INCHES
2. DIMENSIONS ARE IN $\frac{\text{INCHES}}{(\text{MILLIMETERS})}$
3. DRAWING NOT TO SCALE

*DIMENSION DOES NOT INCLUDE MOLD FLASH. MOLD FLASH SHALL NOT EXCEED 0.006" (0.152mm) PER SIDE

**DIMENSION DOES NOT INCLUDE INTERLEAD FLASH. INTERLEAD FLASH SHALL NOT EXCEED 0.010" (0.254mm) PER SIDE

TYPICAL APPLICATION

A 2:1 PGA MUX



MUX OPERATION: IF THE LOWER NIBBLE (Q3, Q2, Q1, Q0) IS (1, 0, 0, 0) THEN OUTA IS IN TRI-STATE AND THE UPPER NIBBLE (Q7, Q6, Q5, Q4) CONTROLS THE ACTIVE CHANNEL B. IF THE UPPER NIBBLE IS (1, 0, 0, 0) THEN OUTB IS IN TRI-STATE AND THE LOWER NIBBLE CONTROLS ACTIVE CHANNEL A.

RELATED PARTS

PART NUMBER	DESCRIPTION	COMMENTS
LT1228	100MHZ Gain Controlled Transconductance Amplifier	Differential Input, Continuous Analog Gain Control
LT1251/LT1256	40Mhz Video Fader and Gain Controlled Amplifier	Two Input, One Output, Continuous Analog Gain Control
LTC1564	10kHz to 150kHz Digitally Controlled Filter and PGA	Continuous Time, Low Noise 8th Order Filter and 4-Bit PGA
LTC6910-1/-2/-3	Digitally Controlled Programmable Gain Amplifier in SOT-23	Single Programmable Gain Amplifier, 3-Bit Parallel Digital Interface
LTC6911-1/-2	Dual Digitally Controlled Programmable Gain Amplifier in MSOP-10	Dual Programmable Gain Amplifiers, 3-Bit Parallel Digital Interface
LTC6915	Zero Drift Instrumentation Amp with Digitally Programmable Gain	Gains 0 - 4096V/V, 116dB CMRR

Linear Technology Corporation
1630 McCarthy Blvd., Milpitas, CA 95035-7417
(408) 432-1900 • FAX: (408) 434-0507 • www.linear.com
LT/LT 1005 REV A • PRINTED IN USA

TEXAS INSTRUMENTS

OPA354, OPA2354, OPA4354

SBOS233F – MARCH 2002 – REVISED JUNE 2016

OPAx354 250-MHz, Rail-to-Rail I/O, CMOS Operational Amplifiers

1 Features

- Unity-Gain Bandwidth: 250 MHz
- Wide Bandwidth: 100-MHz GBW
- High Slew Rate: 150 V/µs
- Low Noise: 6.5 nV/√Hz
- Rail-to-Rail I/O
- High Output Current: > 100 mA
- Excellent Video Performance:
  - Diff Gain: 0.02%, Diff Phase: 0.09°
  - 0.1-dB Gain Flatness: 40 MHz
- Low Input Bias Current: 3 pA
- Quiescent Current: 4.9 mA
- Thermal Shutdown
- Supply Range: 2.5 V to 5.5 V
- *Micro*SIZE and PowerPAD™ Packages

2 Applications

- Video Processing
- Ultrasound
- Optical Networking, Tunable Lasers
- Photodiode Transimpedance Amps
- Active Filters
- High-Speed Integrators
- Analog-to-Digital (A/D) Converter Input Buffers
- Digital-to-Analog (D/A) Converter Output Amplifiers
- Barcode Scanners
- Communications

3 Description

The OPA354 series of high-speed, voltage-feedback CMOS operational amplifiers are designed for video and other applications requiring wide bandwidth. They are unity-gain stable and can drive large output currents. Differential gain is 0.02% and differential phase is 0.09°. Quiescent current is only 4.9 mA per channel.

The OPA354 series op amps are optimized for operation on single or dual supplies as low as 2.5 V (±1.25 V) and up to 5.5 V (±2.75 V). Common-mode input range extends beyond the supplies. The output swing is within 100 mV of the rails, supporting wide dynamic range.

For applications requiring the full 100-mA continuous output current, single and dual 8-pin HSOP PowerPAD versions are available.

The single version (OPA354) is available in the tiny 5-pin SOT-23 and 8-pin HSOP PowerPAD packages. The dual version (OPA2354) comes in the miniature 8-pin VSSOP and 8-pin HSOP PowerPAD packages. The quad version (OPA4354) is offered in 14-pin TSSOP and 14-pin SOIC packages.

Multichannel version feature completely independent circuitry for lowest crosstalk and freedom from interaction. All are specified over the extended –40°C to 125°C temperature range.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
OPA354	HSOP (8)	4.89 mm × 3.90 mm
	SOT-23 (5)	2.90 mm × 1.60 mm
OPA2354	VSSOP (8)	3.00 mm × 3.00 mm
	HSOP (8)	4.89 mm × 3.90 mm
OPA4354	SOIC (14)	8.65 mm × 3.91 mm
	TSSOP (14)	5.00 mm × 4.40 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Simplified Schematic

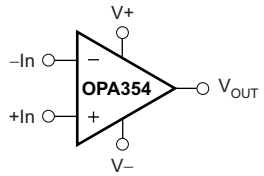


Table of Contents

4 Revision History

Changes from Revision E (March 2002) to Revision F **Page**

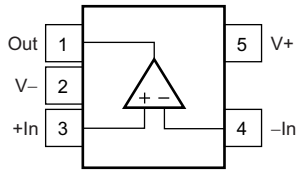
- Added *ESD Ratings* table, *Feature Description* section, *Device Functional Modes*, *Application and Implementation* section, *Power Supply Recommendations* section, *Layout* section, *Device and Documentation Support* section, and *Mechanical, Packaging, and Orderable Information* section ... 1
- Deleted *Package/Ordering Information* table, see POA at the end of the data sheet ... 1
- Renamed *OPAx354 Related Products* table to *Device Comparison Table* ... 3

5 Device Comparison Table

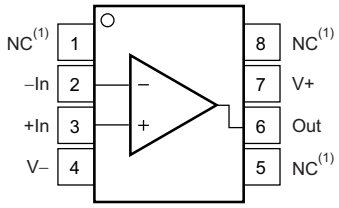
FEATURES	PRODUCT
Shutdown Version of OPA354 Family	OPAx357
200-MHz GBW, Rail-to-Rail Output, CMOS, Shutdown	OPAx355
200-MHz GBW, Rail-to-Rail Output, CMOS	OPAx356
38-MHz GBW, Rail-to-Rail Input/Output, CMOS	OPAx350/OPAx353
75-MHz BW G = 2, Rail-to-Rail Output	OPA2631
150-MHz BW G = 2, Rail-to-Rail Output	OPA2634
100-MHz BW, Differential Input/Output, 3.3-V Supply	THS412x

6 Pin Configuration and Functions

OPA354: DBV Package
5-Pin SOT-23
Top View



OPA354: DDA Package
8-Pin HSOP(2)
Top View



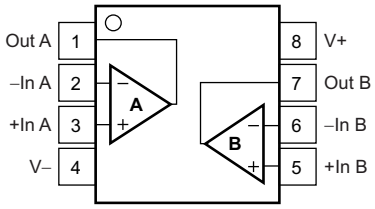
(1) NC means no internal connection.

PowerPAD must be connected to V− or left floating.

Pin Functions: OPA354

PIN			I/O	DESCRIPTION
NAME	SOT-23	HSOP		
–In	4	2	I	Inverting input
+In	3	3	I	Noninverting input
NC	—	1, 5, 8	—	No internal connection (can be left floating)
Out	1	6	O	Output
V–	2	4	—	Negative (lowest) supply
V+	5	7	—	Positive (highest) supply

OPA2354: DGK and DDA Packages
8-Pin VSSOP, HSOP(1)
Top View

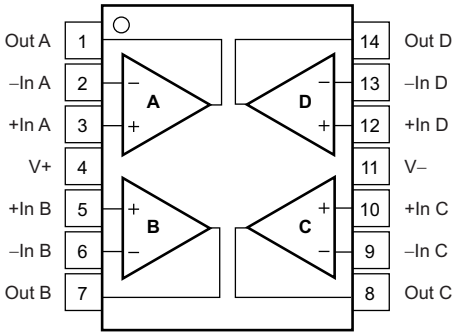


(1) PowerPAD must be connected to V– or left floating.

Pin Functions: OPA2354

PIN			I/O	DESCRIPTION
NAME	VSSOP	HSOP		
–In A	2	2	I	Inverting input, channel A
+In A	3	3	I	Noninverting input, channel A
–In B	6	6	I	Inverting input, channel B
+In B	5	5	I	Noninverting input, channel B
Out A	1	1	O	Output, channel A
Out B	7	7	O	Output, channel B
V–	4	4	—	Negative (lowest) supply
V+	8	8	—	Positive (highest) supply

OPA4354: D and PW Packages
14-Pin SOIC, TSSOP
Top View



Pin Functions: OPA4354

PIN			I/O	DESCRIPTION
NAME	SOIC	TSSOP		
–In A	2	2	I	Inverting input, channel A
+In A	3	3	I	Noninverting input, channel A
–In B	6	6	I	Inverting input, channel B
+In B	5	5	I	Noninverting input, channel B
–In C	9	9	I	Inverting input, channel C
+In C	10	10	I	Noninverting input, channel C
–In D	13	13	I	Inverting input, channel D
+In D	12	12	I	Noninverting input, channel D
Out A	1	1	O	Output, channel A
Out B	7	7	O	Output, channel B
Out C	8	8	O	Output, channel C
Out D	14	14	O	Output, channel D
V–	11	11	—	Negative (lowest) supply
V+	4	4	—	Positive (highest) supply

7 Specifications

7.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)

		MIN	MAX	UNIT
Voltage	Supply voltage, V+ to V−		7.5	V
	Signal input terminals(2)	(V−) − (0.5)	(V+) + 0.5	
Current	Signal input terminals(2)	–10	10	mA
	Output short circuit(3)	Continuous		
Temperature	Operating, T_A	–55	150	°C
	Junction, T_J		150	
	Storage, T_{stg}	–65	150	

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) Input pins are diode-clamped to the power-supply rails. Input signals that can swing more than 0.5 V beyond the supply rails must be current limited to 10 mA or less.
(3) Short-circuit to ground, one amplifier per package.

7.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±2000	V
		Charged-device model (CDM), per JEDEC specification JESD22-C101(2)	±250	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

7.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

		MIN	MAX	UNIT
V_S	Supply voltage, V– to V+	2.5	5.5	V
	Specified temperature	–40	125	°C

7.4 Thermal Information: OPA354

THERMAL METRIC(1)		OPA354		UNIT
		DBV (SOT-23)	DDA (HSOP)	
		5 PINS	8 PINS	
$R_{\theta JA}$	Junction-to-ambient thermal resistance	216.3	42.5	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	84.3	54	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	43.1	26.5	°C/W
ψ_{JT}	Junction-to-top characterization parameter	3.8	8	°C/W
ψ_{JB}	Junction-to-board characterization parameter	42.3	26.4	°C/W
$R_{\theta JC(bot)}$	Junction-to-case (bottom) thermal resistance	—	3.6	°C/W

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report.

7.5 Thermal Information: OPA2354

THERMAL METRIC(1)		OPA2354		UNIT
		DDA (HSOP)	DGK (VSSOP)	
		8 PINS	8 PINS	
$R_{\theta JA}$	Junction-to-ambient thermal resistance	40.6	175.9	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	46	67.8	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	20.7	97.1	°C/W
ψ_{JT}	Junction-to-top characterization parameter	5.6	9.3	°C/W
ψ_{JB}	Junction-to-board characterization parameter	20.6	95.5	°C/W
$R_{\theta JC(bot)}$	Junction-to-case (bottom) thermal resistance	2.5	—	°C/W

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report.

7.6 Thermal Information: OPA4354

THERMAL METRIC(1)		OPA4354		UNIT
		D (SOIC)	PW (TSSOP)	
		14 PINS	14 PINS	
$R_{\theta JA}$	Junction-to-ambient thermal resistance	83.8	92.6	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	70.7	27.5	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	59.5	33.6	°C/W
ψ_{JT}	Junction-to-top characterization parameter	11.6	1.9	°C/W
ψ_{JB}	Junction-to-board characterization parameter	37.7	33.1	°C/W
$R_{\theta JC(bot)}$	Junction-to-case (bottom) thermal resistance	—	—	°C/W

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report.

7.7 Electrical Characteristics: V_S = 2.7 V to 5.5 V Single-Supply

At T_A = 25°C, R_F = 0 Ω, R_L = 1 kΩ, and connected to V_S/2, unless otherwise noted.

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
OFFSET VOLTAGE						
V_{OS}	Input offset voltage	V_S = 5 V, at T_A = 25°C		±2	±8	mV
		V_S = 5 V, at T_A = −40°C to 125°C			±10	
dV_{OS}/dT	Input offset voltage vs temperature	V_S = 5 V, at T_A = −40°C to 125°C		±4		µV/°C
PSRR	Input offset voltage vs power supply	V_S = 2.7 V to 5.5 V, V_{CM} = (V_S/2) − 0.55 V		±200	±800	µV/V
		V_S = 2.7 V to 5.5 V, V_{CM} = (V_S/2) − 0.55 V, at T_A = −40°C to 125°C			±900	
INPUT BIAS CURRENT						
I_B	Input bias current			3	±50	pA
I_{OS}	Input offset current			±1	±50	pA
NOISE						
e_n	Input voltage noise density	f = 1 MHz		6.5		nV/√Hz
i_n	Current noise density	f = 1 MHz		50		fA/√Hz
INPUT VOLTAGE RANGE						
V_{CM}	Common-mode voltage		(V−) − 0.1		(V+) + 0.1	V
CMRR	Common-mode rejection ratio	V_S = 5.5 V, –0.1 V < V_{CM} < 3.5 V, at T_A = 25°C	66	80		dB
		V_S = 5.5 V, –0.1 V < V_{CM} < 3.5 V, at T_A = −40°C to 125°C	64			
		V_S = 5.5 V, –0.1 V < V_{CM} < 5.6 V, at T_A = 25°C	56	68		
		V_S = 5.5 V, –0.1 V < V_{CM} < 5.6 V, at T_A = −40°C to 125°C	55			
INPUT IMPEDANCE						
	Differential			10^{13} || 2		Ω || pF
	Common-mode			10^{13} || 2		Ω || pF
OPEN-LOOP GAIN						
A_{OL}	Open-loop gain	V_S = 5.5 V, 0.3 V < V_O < 4.7 V, at T_A = 25°C	94	110		dB
		V_S = 5 V, 0.4 V < V_O < 4.6 V, at T_A = −40°C to 125°C	90			
FREQUENCY RESPONSE						
f_{-3dB}	Small-signal bandwidth	At G = +1, V_O = 100 mV_{PP}, R_F = 25 Ω		250		MHz
		At G = +2, V_O = 100 mV_{PP}		90		
GBW	Gain-bandwidth product	G = +10		100		MHz
$f_{0.1dB}$	Bandwidth for 0.1-dB gain flatness	At G = +2, V_O = 100 mV_{PP}		40		MHz
SR	Slew rate	V_S = 5 V, G = +1, 4-V step		150		V/µs
		V_S = 5 V, G = +1, 2-V step		130		
		V_S = 3 V, G = +1, 2-V step		110		
	Rise-and-fall time	At G = +1, V_O = 200 mV_{PP}, 10% to 90%		2		ns
		At G = +1, V_O = 2 V_{PP}, 10% to 90%		11		
	Settling time	0.1%, V_S = 5 V, G = +1, 2-V output step		30		ns
		0.01%, V_S = 5 V, G = +1, 2-V output step		60		
	Overload recovery time	V_{IN} × Gain = V_S		5		ns

Electrical Characteristics: V_S = 2.7 V to 5.5 V Single-Supply (continued)

At T_A = 25°C, R_F = 0 Ω, R_L = 1 kΩ, and connected to V_S/2, unless otherwise noted.

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
FREQUENCY RESPONSE, continued							
	Harmonic distortion	Second harmonic	At G = +1, f = 1 MHz, V_O = 2 V_{PP}, R_L = 200 Ω, V_{CM} = 1.5 V		–75		dBc
		Third harmonic	At G = +1, f = 1 MHz, V_O = 2 V_{PP}, R_L = 200 Ω, V_{CM} = 1.5 V		–83		
	Differential gain error		NTSC, R_L = 150 Ω		0.02%		
	Differential phase error		NTSC, R_L = 150 Ω		0.09		°
	Channel-to-channel crosstalk	OPA2354	f = 5 MHz		–100		dB
		OPA4354			–84		
OUTPUT							
	Voltage output swing from rail		V_S = 5 V, R_L = 1 kΩ, A_{OL} > 94 dB, at T_A = 25°C		0.1	0.3	V
			V_S = 5 V, R_L = 1 kΩ, A_{OL} > 90 dB, at T_A = −40°C to 125°C			0.4	
I_O	Output current, single, dual, quad[1][2]		V_S = 5 V	100			mA
			V_S = 3 V		50		mA
	Closed-loop output impedance		f < 100 kHz		0.05		Ω
R_O	Open-loop output resistance				35		Ω
POWER SUPPLY							
V_S	Specified voltage			2.7		5	V
	Operating voltage			2.5		5.5	
I_Q	Quiescent current (per amplifier)		At T_A = 25°C, V_S = 5 V, enabled, I_O = 0		4.9	6	mA
			At T_A = –40°C to 125°C			7.5	
THERMAL SHUTDOWN – JUNCTION TEMPERATURE							
	Shutdown				160		°C
	Reset from shutdown				140		°C
THERMAL RANGE							
	Specified			–40		125	°C
	Operating			–55		150	°C
	Storage			–65		150	°C

(1) See typical characteristic curves, *Output Voltage Swing vs Output Current* (Figure 20 and Figure 22).
(2) Specified by design.

7.8 Typical Characteristics

At T_A = 25°C, V_S = 5 V, G = +1, R_F = 0 Ω, R_L = 1 kΩ, and connected to V_S/2, unless otherwise noted.

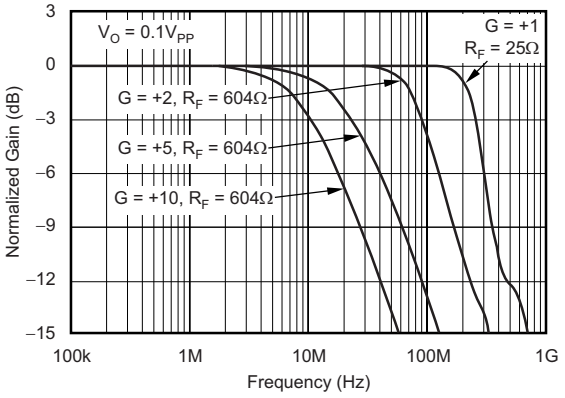


Figure 1. Noninverting Small-Signal Frequency Response

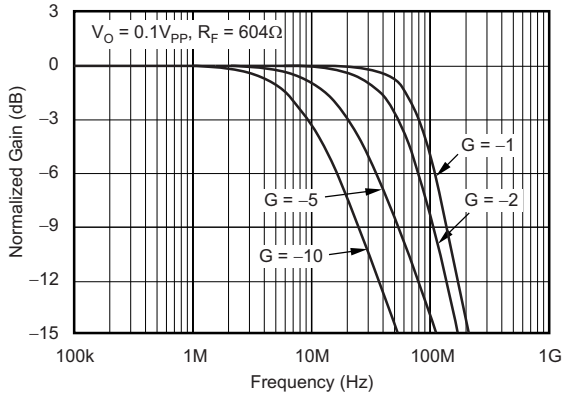


Figure 2. Inverting Small-Signal Frequency Response

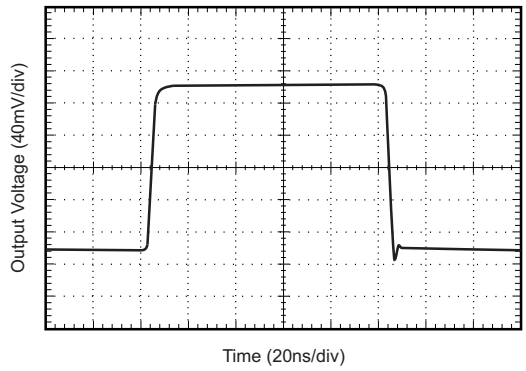


Figure 3. Noninverting Small-Signal Step Response

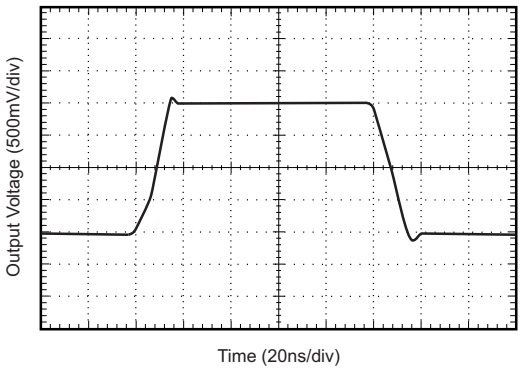


Figure 4. Noninverting Large-Signal Step Response

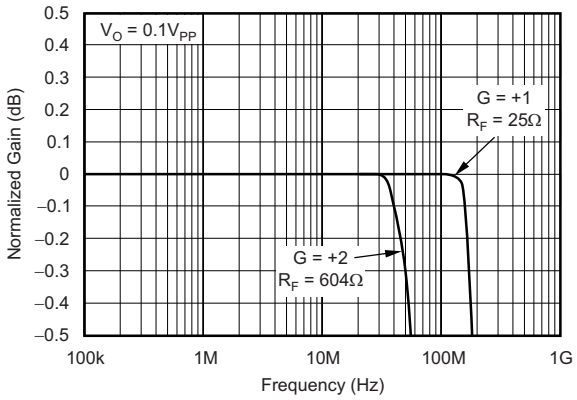


Figure 5. 0.1-dB Gain Flatness

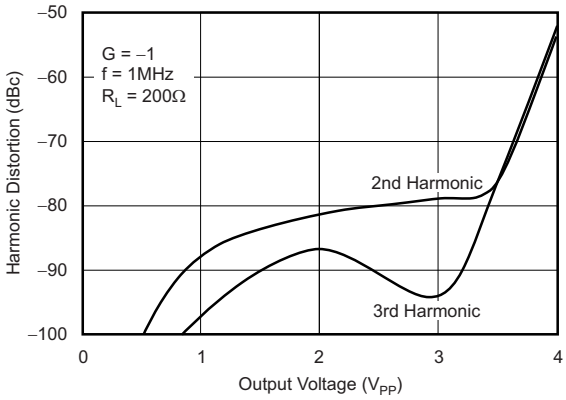


Figure 6. Harmonic Distortion vs Output Voltage

Typical Characteristics (continued)

At T_A = 25°C, V_S = 5 V, G = +1, R_F = 0 Ω, R_L = 1 kΩ, and connected to V_S/2, unless otherwise noted.

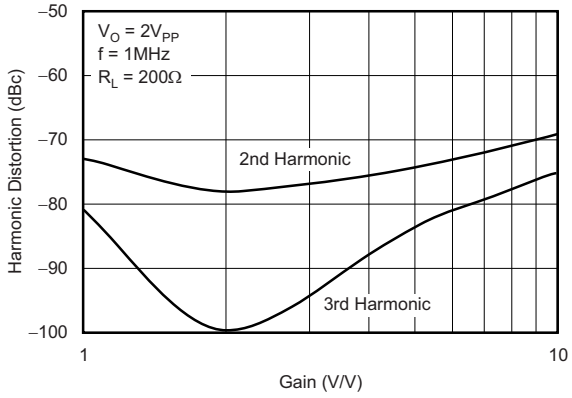


Figure 7. Harmonic Distortion vs Noninverting Gain

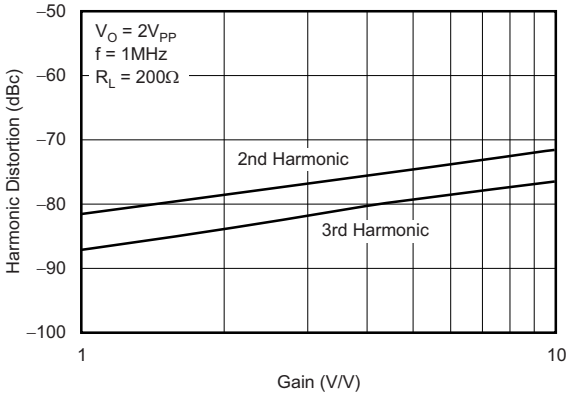


Figure 8. Harmonic Distortion vs Inverting Gain

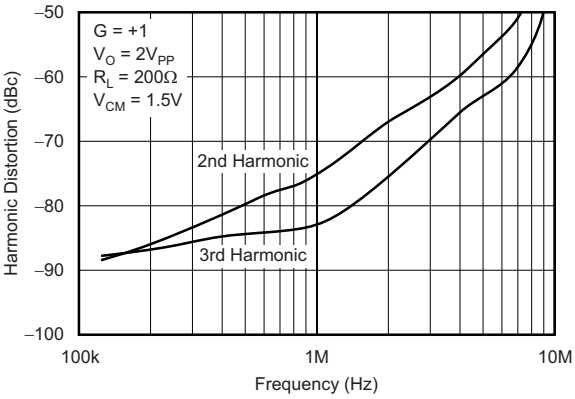


Figure 9. Harmonic Distortion vs Frequency

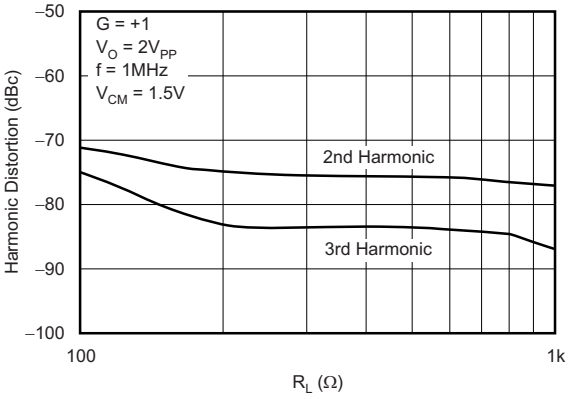


Figure 10. Harmonic Distortion vs Load Resistance

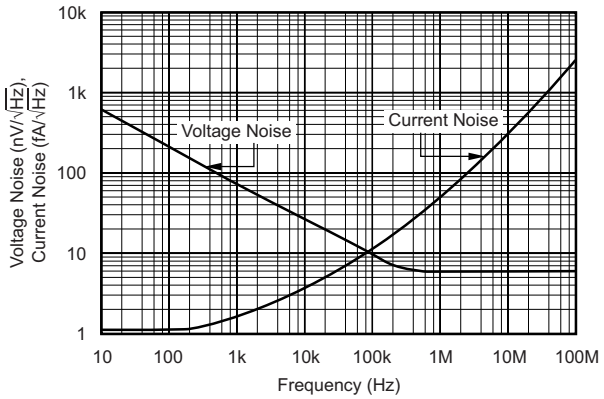


Figure 11. Input Voltage and Current Noise Spectral Density vs Frequency

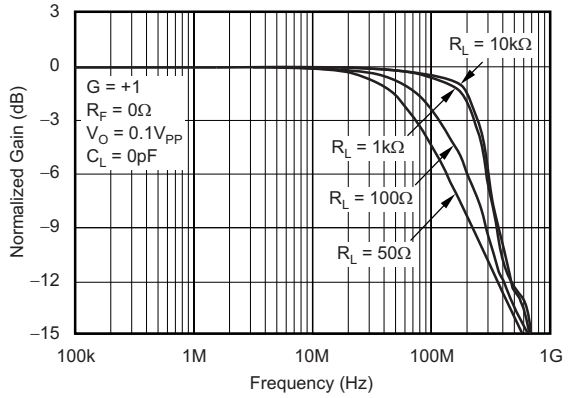


Figure 12. Frequency Response for Various R_L

Typical Characteristics (continued)

At T_A = 25°C, V_S = 5 V, G = +1, R_F = 0 Ω, R_L = 1 kΩ, and connected to V_S/2, unless otherwise noted.

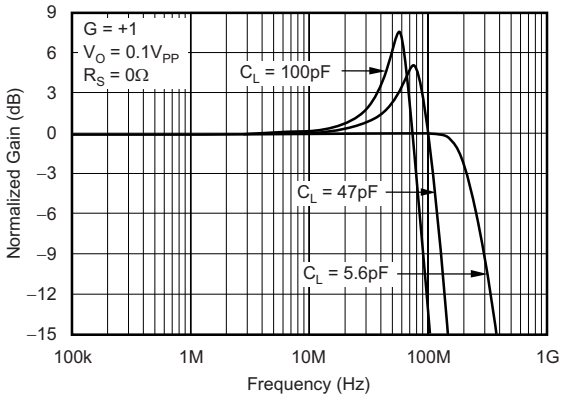


Figure 13. Frequency Response for Various C_L

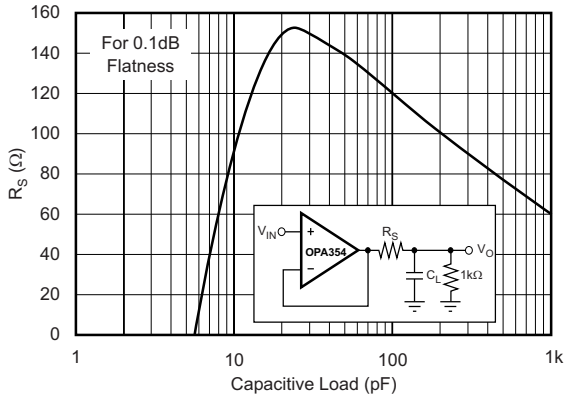


Figure 14. Recommended R_S vs Capacitive Load

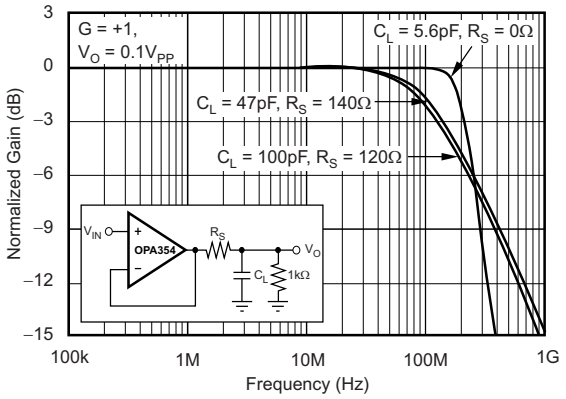


Figure 15. Frequency Response vs Capacitive Load

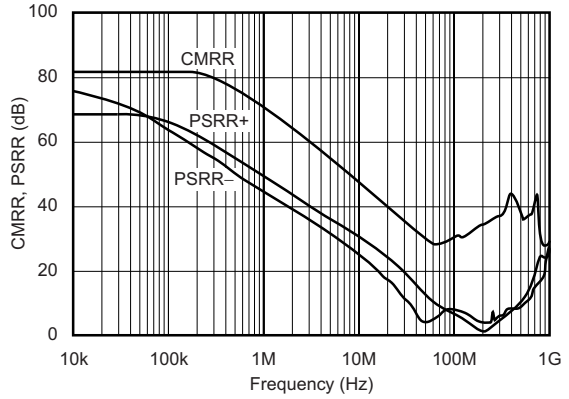


Figure 16. Common-Mode Rejection Ratio and Power-Supply Rejection Ratio vs Frequency

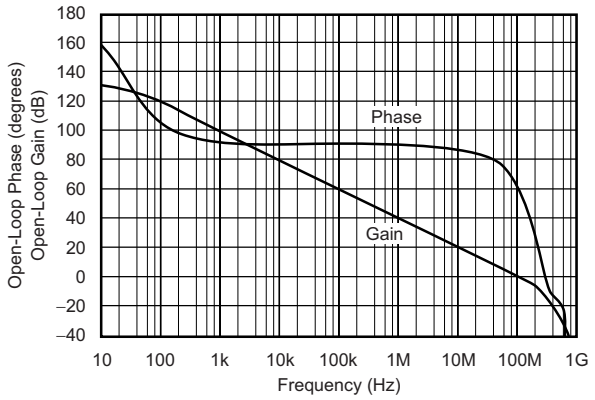


Figure 17. Open-Loop Gain and Phase

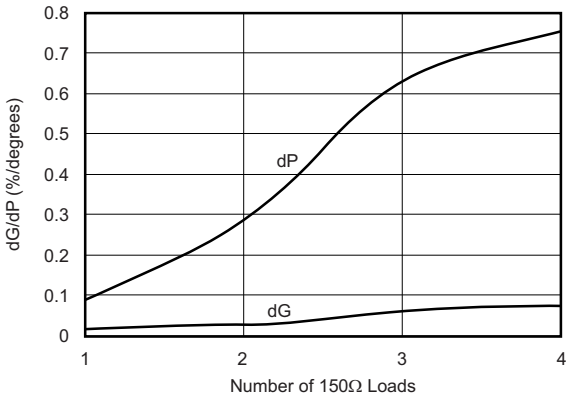


Figure 18. Composite Video Differential Gain and Phase

Typical Characteristics (continued)

At T_A = 25°C, V_S = 5 V, G = +1, R_F = 0 Ω, R_L = 1 kΩ, and connected to V_S/2, unless otherwise noted.

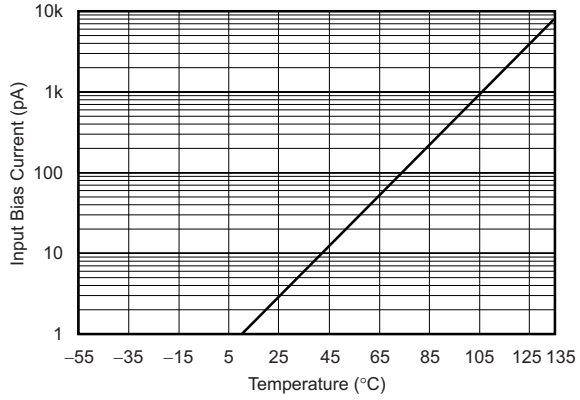


Figure 19. Input Bias Current vs Temperature

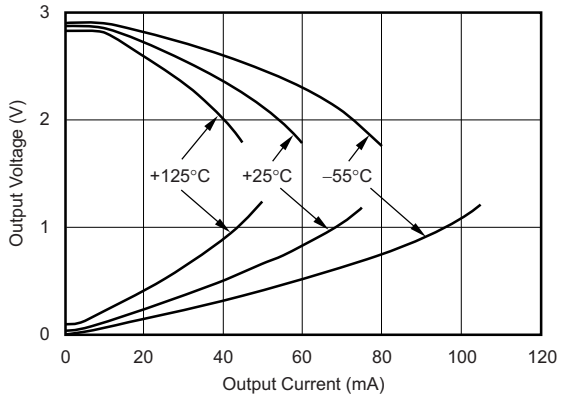


Figure 20. Output Voltage Swing vs Output Current for V_S = 3 V

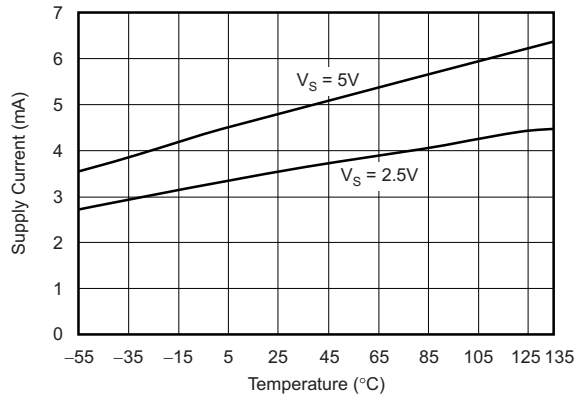


Figure 21. Supply Current vs Temperature

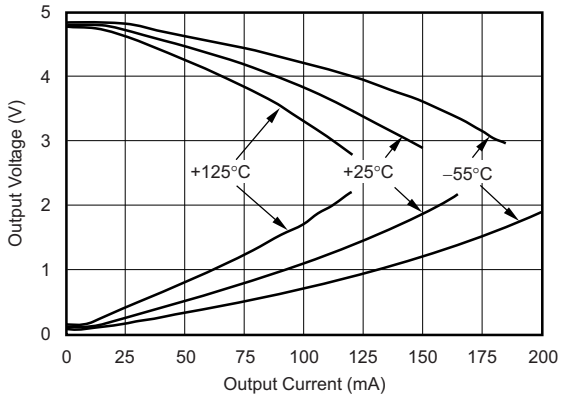


Figure 22. Output Voltage Swing vs Output Current for V_S = 5 V

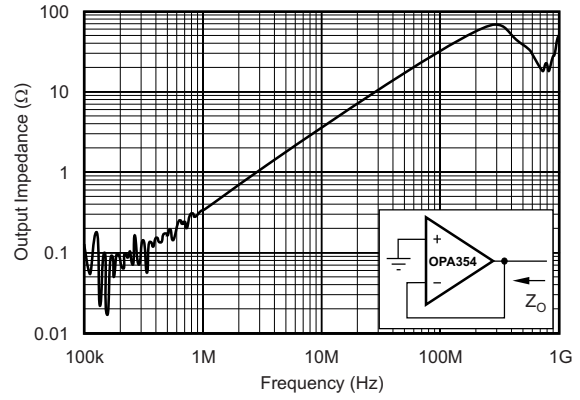


Figure 23. Closed-Loop Output Impedance vs Frequency

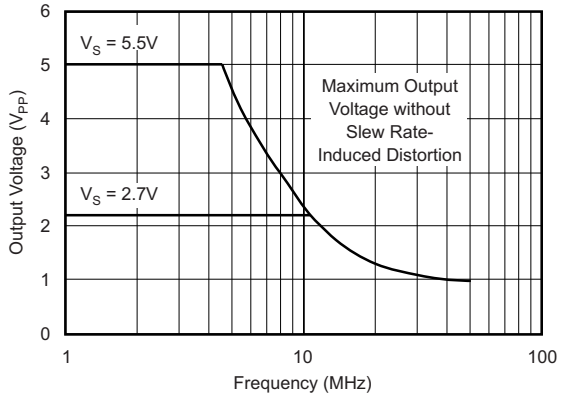


Figure 24. Maximum Output Voltage vs Frequency

Typical Characteristics (continued)

At T_A = 25°C, V_S = 5 V, G = +1, R_F = 0 Ω, R_L = 1 kΩ, and connected to V_S/2, unless otherwise noted.

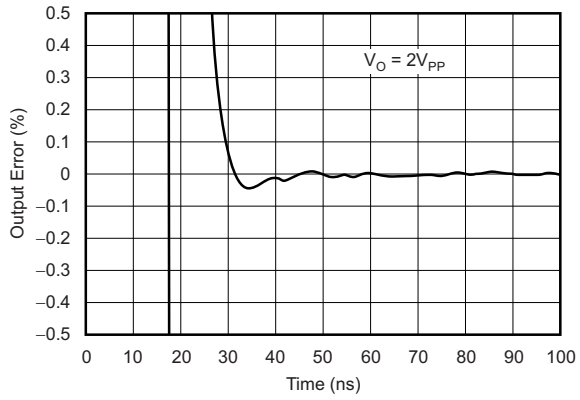


Figure 25. Output Settling Time to 0.1%

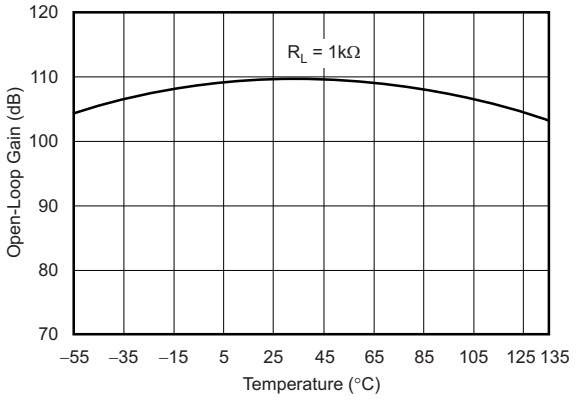


Figure 26. Open-Loop Gain vs Temperature

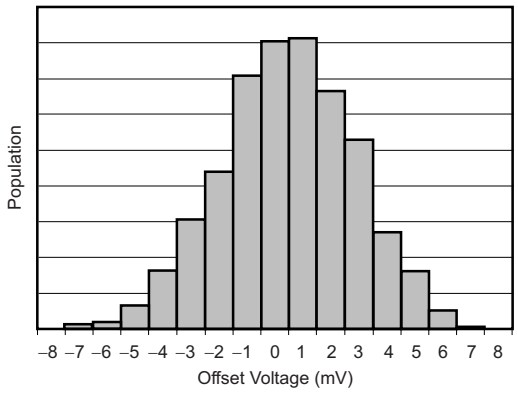


Figure 27. Offset Voltage Production Distribution

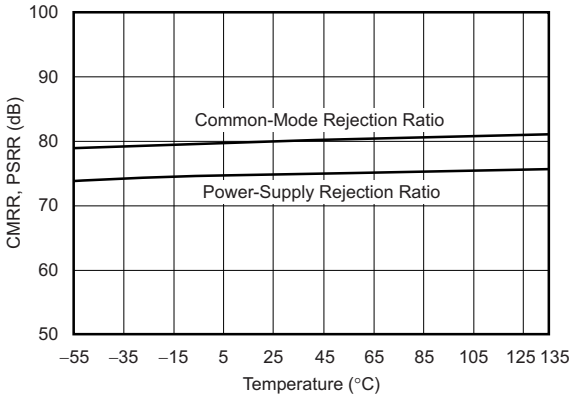


Figure 28. Common-Mode Rejection Ratio and Power-Supply Rejection Ratio vs Temperature

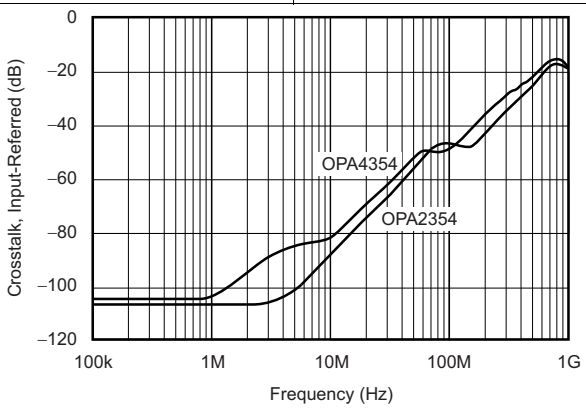


Figure 29. Channel-to-Channel Crosstalk

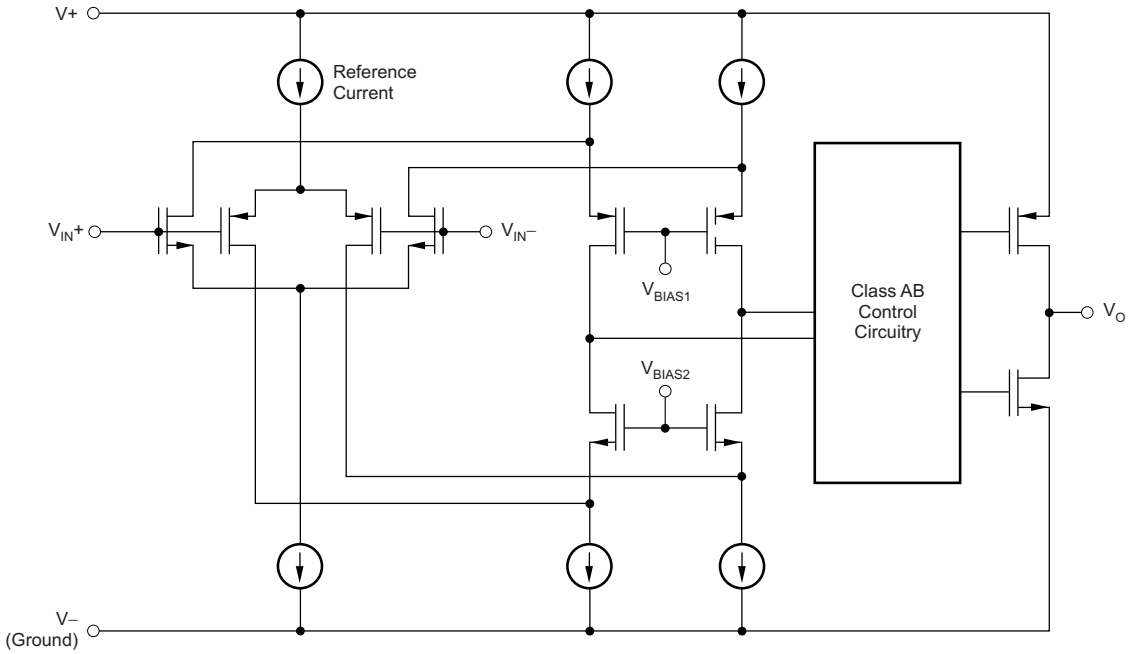
8 Detailed Description

8.1 Overview

The OPA354 is a CMOS, rail-to-rail I/O, high-speed, voltage-feedback operational amplifier designed for video, high-speed, and other applications. It is available as a single, dual, or quad op amp.

The amplifier features a 100-MHz gain bandwidth, and 150-V/µs slew rate, but it is unity-gain stable and can be operated as a +1-V/V voltage follower.

8.2 Functional Block Diagram



8.3 Feature Description

8.3.1 Operating Voltage

The OPA354 is specified over a power-supply range of 2.7 V to 5.5 V (±1.35 V to ±2.75 V). However, the supply voltage may range from 2.5 V to 5.5 V (±1.25 V to ±2.75 V). Supply voltages higher than 7.5 V (absolute maximum) can permanently damage the amplifier.

Parameters that vary over supply voltage or temperature are shown in *Typical Characteristics* of this data sheet.

8.3.2 Rail-to-Rail Input

The specified input common-mode voltage range of the OPA354 extends 100 mV beyond the supply rails. This extended range is achieved with a complementary input stage—an N-channel input differential pair in parallel with a P-channel differential pair, as shown in the *Functional Block Diagram*. The N-channel pair is active for input voltages close to the positive rail, typically (V+) − 1.2 V to 100 mV above the positive supply, while the P-channel pair is on for inputs from 100 mV below the negative supply to approximately (V+) − 1.2 V. There is a small transition region, typically (V+) − 1.5 V to (V+) − 0.9 V, in which both pairs are on. This 600-mV transition region can vary ±500 mV with process variation. Thus, the transition region (both input stages on) can range from (V+) − 2 V to (V+) − 1.5 V on the low end, up to (V+) − 0.9 V to (V+) − 0.4 V on the high end.

A double-folded cascode adds the signal from the two input pairs and presents a differential signal to the class AB output stage.

8.3.3 Rail-to-Rail Output

A class AB output stage with common-source transistors is used to achieve rail-to-rail output. For high-impedance loads (> 200 Ω), the output voltage swing is typically 100 mV from the supply rails. With 10-Ω loads, a useful output swing can be achieved while maintaining high open-loop gain. See the typical characteristic curves, *Output Voltage Swing vs Output Current* (Figure 20 and Figure 22).

8.3.4 Output Drive

The OPA354 output stage can supply a continuous output current of ±100 mA and yet provide approximately 2.7 V of output swing on a 5-V supply, as shown in Figure 30. For maximum reliability, TI does not recommend running a continuous DC current in excess of ±100 mA. Refer to the typical characteristic curves, *Output Voltage Swing vs Output Current* (Figure 20 and Figure 22). For supplying continuous output currents greater than ±100 mA, the OPA354 may be operated in parallel, as shown in Figure 31.

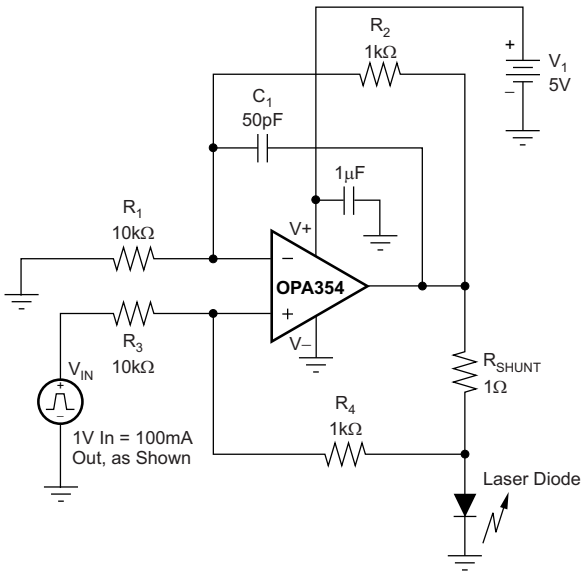


Figure 30. Laser Diode Driver

Feature Description (continued)

The OPA354 provides peak currents up to 200 mA, which corresponds to the typical short-circuit current. Therefore, an on-chip thermal shutdown circuit is provided to protect the OPA354 from dangerously high junction temperatures. At 160°C, the protection circuit shuts down the amplifier. Normal operation resumes when the junction temperature cools to below 140°C.

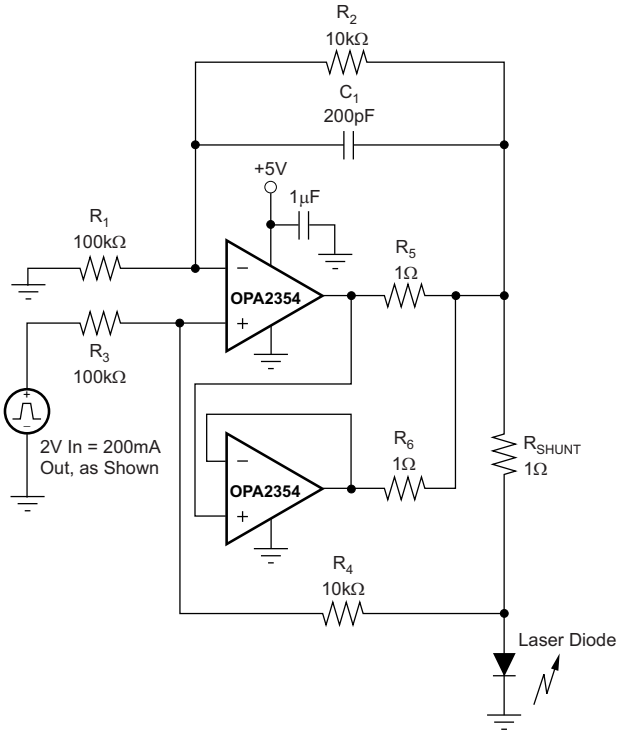


Figure 31. Parallel Operation

8.3.5 Video

The OPA354 output stage is capable of driving standard back-terminated 75-Ω video cables, as shown in Figure 32. By back-terminating a transmission line, it does not exhibit a capacitive load to its driver. A properly back-terminated 75-Ω cable does not appear as capacitance; it presents only a 150-Ω resistive load to the OPA354 output.

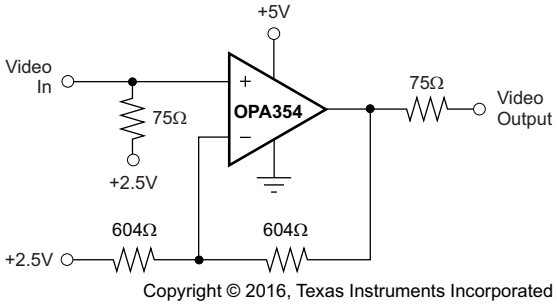
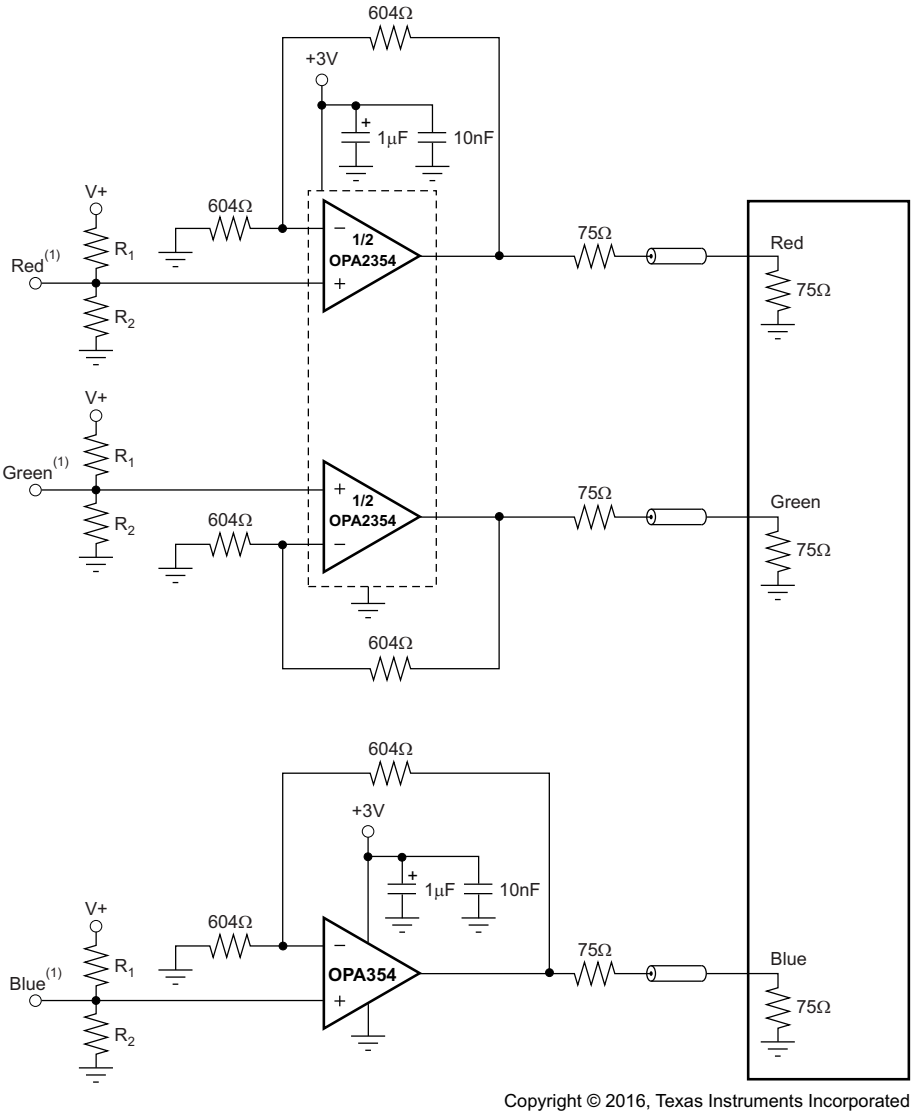


Figure 32. Single-Supply Video Line Driver

The OPA354 can be used as an amplifier for RGB graphic signals, which have a voltage of zero at the video black level, by offsetting and AC-coupling the signal. See Figure 33.

Feature Description (continued)



(1) Source video signal offset 300 mV above ground to accommodate op amp swing−to−ground capability.

Figure 33. RGB Cable Driver

Feature Description (continued)

8.3.6 Driving Analog-to-Digital converters

The OPA354 series op amps offer 60 ns of settling time to 0.01%, making them a good choice for driving high- and medium-speed sampling A/D converters and reference circuits. The OPA354 series provide an effective means of buffering the A/D converter input capacitance and resulting charge injection while providing signal gain. For applications requiring high DC accuracy, the OPA350 series is recommended.

Figure 34 illustrates the OPA354 driving an A/D converter. With the OPA354 in an inverting configuration, a capacitor across the feedback resistor can be used to filter high-frequency noise in the signal.

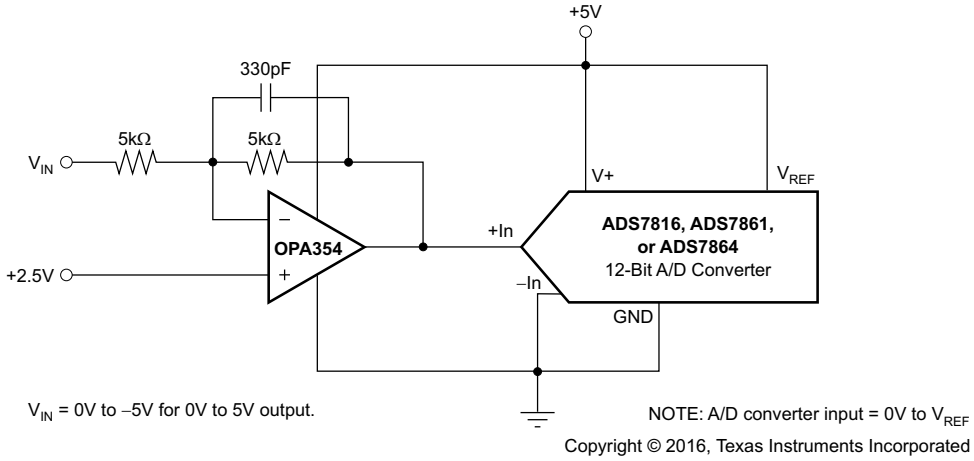


Figure 34. The OPA354 in Inverting Configuration Driving the ADS7816

8.3.7 Capacitive Load and Stability

The OPA354 series op amps can drive a wide range of capacitive loads. However, all op amps under certain conditions may become unstable. Op amp configuration, gain, and load value are just a few of the factors to consider when determining stability. An op amp in unity-gain configuration is most susceptible to the effects of capacitive loading. The capacitive load reacts with the device output resistance, along with any additional load resistance, to create a pole in the small-signal response that degrades the phase margin. Refer to the typical characteristic curve, *Frequency Response for Various C_L* (Figure 13) for details.

The OPA354 topology enhances its ability to drive capacitive loads. In unity gain, these op amps perform well with large capacitive loads. Refer to the typical characteristic curves, *Recommended R_S vs Capacitive Load* (Figure 14) and *Frequency Response vs Capacitive Load* (Figure 15) for details.

One method of improving capacitive load drive in the unity-gain configuration is to insert a 10-Ω to 20-Ω resistor in series with the output, as shown in Figure 35. This configuration significantly reduces ringing with large capacitive loads—see the typical characteristic curve, *Frequency Response vs Capacitive Load* (Figure 15). However, if there is a resistive load in parallel with the capacitive load, R_S creates a voltage divider. This voltage division introduces a DC error at the output and slightly reduces output swing. This error may be insignificant. For instance, with R_L = 10 kΩ and R_S = 20 Ω, there is approximately a 0.2% error at the output.

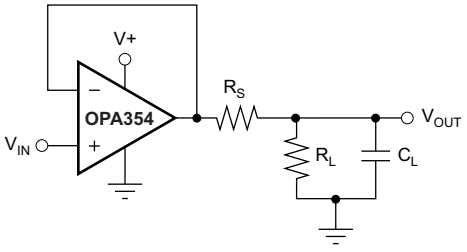


Figure 35. Series Resistor in Unity-Gain Configuration Improves Capacitive Load Drive

Feature Description (continued)

8.3.8 Wideband Transimpedance Amplifier

Wide bandwidth, low input bias current, low input voltage, and current noise make the OPA354 an ideal wideband photodiode transimpedance amplifier for low-voltage single-supply applications. Low-voltage noise is important because photodiode capacitance causes the effective noise gain of the circuit to increase at high frequency.

The key elements to a transimpedance design, as shown in Figure 36, are the expected diode capacitance [including the parasitic input common-mode and differential-mode input capacitance (2 + 2) pF for the OPA354], the desired transimpedance gain (R_F), and the Gain-Bandwidth Product (GBW) for the OPA354 (100 MHz typical). With these three variables set, the feedback capacitor value (C_F) may be set to control the frequency response.

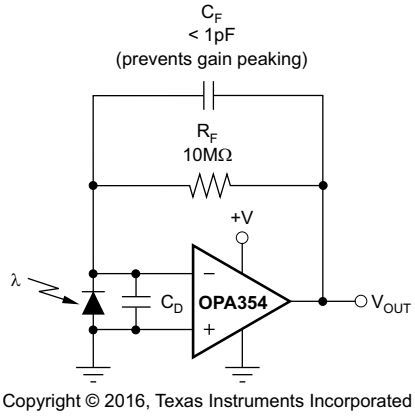


Figure 36. Transimpedance Amplifier

To achieve a maximally flat, second-order, Butterworth frequency response, the feedback pole must be set as shown in Equation 1:

$$\frac{1}{2\pi R_F C_F} = \sqrt{\frac{GBP}{4\pi R_F C_D}} \quad (1)$$

Typical surface-mount resistors have a parasitic capacitance of approximately 0.2 pF that must be deducted from the calculated feedback capacitance value. Bandwidth is calculated by Equation 2:

$$f_{-3dB} = \sqrt{\frac{GBP}{2\pi R_F C_D}} \text{ Hz} \quad (2)$$

For even higher transimpedance bandwidth, the high-speed CMOS OPA355 (200-MHz GBW) or the OPA655 (400-MHz GBW) may be used.

8.4 Device Functional Modes

The OPAx354 family of devices is powered on when the supply is connected. The devices can be operated as single-supply operational amplifiers or dual-supply amplifiers depending on the application. The devices can also be used with asymmetrical supplies as long as the differential voltage (V– to V+) is at least 1.8 V and no greater than 5.5 V (example: V– set to –3.5 V and V+ set to 1.5 V).

9 Application and Implementation

NOTE

Information in the following applications sections is not part of the TI component specification, and TI does not warrant its accuracy or completeness. TI's customers are responsible for determining suitability of components for their purposes. Customers should validate and test their design implementation to confirm system functionality.

9.1 Application Information

The OPAx354 family of devices is a CMOS, rail-to-rail I/O, high-speed, voltage-feedback operational amplifier designed for video, high-speed, and other applications. The OPAx354 family of devices is available as a single, dual, or quad op amp. The amplifier features a 100-MHz gain bandwidth, and 150-V/µs slew rate, but it is unity-gain stable and can be operated as a 1-V/V voltage follower.

9.2 Typical Application

Wide gain bandwidth, low input bias current, low input voltage, and current noise make the OPAx354 family of devices an ideal wideband photodiode transimpedance amplifier. Low-voltage noise is important because photodiode capacitance causes the effective noise gain of the circuit to increase at high frequency. The key elements to a transimpedance design, as shown in Figure 37, are the expected diode capacitance, which include the parasitic input common-mode and differential-mode input capacitance; the desired transimpedance gain; and the gain-bandwidth (GBW) for the OPAx354 family of devices (20 MHz). With these three variables set, the feedback capacitor value can be set to control the frequency response. Feedback capacitance includes the stray capacitance of, which is 0.2 pF for a typical surface-mount resistor.

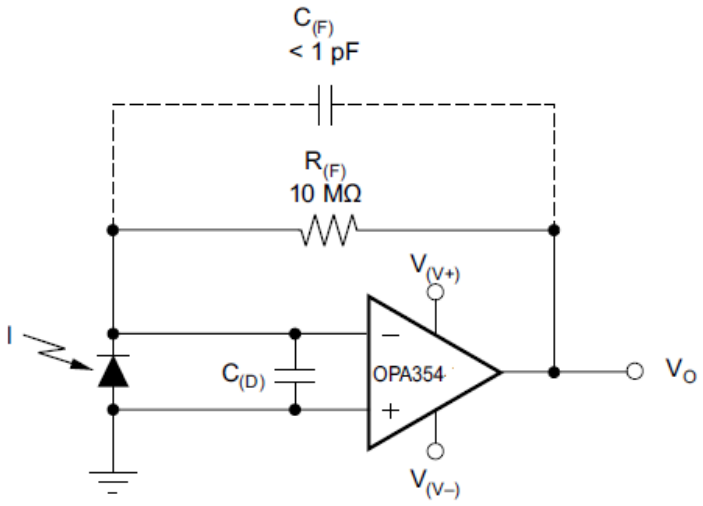


Figure 37. Dual-Supply Transimpedance Amplifier

9.2.1 Design Requirements

For this design example, use the parameters listed in Table 1 as the input parameters.

Table 1. Design Parameters

PARAMETER	EXAMPLE VALUE
Supply voltage, $V_{(V+)}$	2.5 V
Supply voltage, $V_{(V-)}$	–2.5 V

$C_{(F)}$ is optional to prevent gain peaking. $C_{(F)}$ includes the stray capacitance of $R_{(F)}$.

9.2.2 Detailed Design Procedure

To achieve a maximally-flat, second-order Butterworth frequency response, set the feedback pole using Equation 3.

$$\frac{1}{2 \times \pi \times R_{(F)} \times C_{(F)}} = \sqrt{\frac{GBW}{4 \times \pi \times R_{(F)} \times C_{(D)}}} \quad (3)$$

Calculate the bandwidth using Equation 4.

$$f_{(-3\ dB)} = \sqrt{\frac{GBW}{2 \times \pi \times R_{(F)} \times C_{(D)}}} \quad (4)$$

9.2.2.1 Optimizing the Transimpedance Circuit

To achieve the best performance, components must be selected according to the following guidelines:

1. For lowest noise, select $R_{(F)}$ to create the total required gain. Using a lower value for $R_{(F)}$ and adding gain after the transimpedance amplifier generally produces poorer noise performance. The noise produced by $R_{(F)}$ increases with the square-root of $R_{(F)}$, whereas the signal increases linearly. Therefore, signal-to-noise ratio improves when all the required gain is placed in the transimpedance stage.

2. Minimize photodiode capacitance and stray capacitance at the summing junction (inverting input). This capacitance causes the voltage noise of the op amp to be amplified (increasing amplification at high frequency). Using a low-noise voltage source to reverse-bias a photodiode can significantly reduce the capacitance. Smaller photodiodes have lower capacitance. Use optics to concentrate light on a small photodiode.

3. Noise increases with increased bandwidth. Limit the circuit bandwidth to only that required. Use a capacitor across the $R_{(F)}$ to limit bandwidth, even if not required for stability.

4. Circuit board leakage can degrade the performance of an otherwise well-designed amplifier. Clean the circuit board carefully. A circuit board guard trace that encircles the summing junction and is driven at the same voltage can help control leakage.

9.2.3 Application Curve

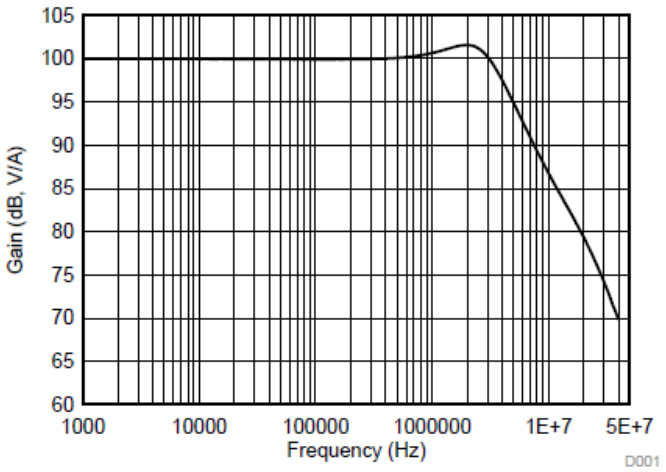


Figure 38. AC Transfer Function

10 Power Supply Recommendations

The OPAx354 family of devices is specified for operation from 2.5 V to 5.5 V (±1.25 to ±2.75 V); many specifications apply from –40°C to 125°C. Parameters that can exhibit significant variance with regard to operating voltage or temperature are shown *Typical Characteristics*.

Place 0.1-µF bypass capacitors close to the power-supply pins to reduce errors coupling in from noisy or high impedance power supplies. For more detailed information on bypass capacitor placement, see *Layout Guidelines*.

11 Layout

11.1 Layout Guidelines

Good high-frequency printed-circuit board (PCB) layout techniques must be employed for the OPA354. Generous use of ground planes, short and direct signal traces, and a suitable bypass capacitor located at the V+ pin assure clean, stable operation. Large areas of copper also provides a means of dissipating heat that is generated in normal operation.

TI does not recommend using sockets with any high-speed amplifier.

A 10-nF ceramic bypass capacitor is the minimum recommended value; adding a 1-µF or larger tantalum capacitor in parallel can be beneficial when driving a low-resistance load. Providing adequate bypass capacitance is essential to achieving very low harmonic and intermodulation distortion.

11.2 Layout Example

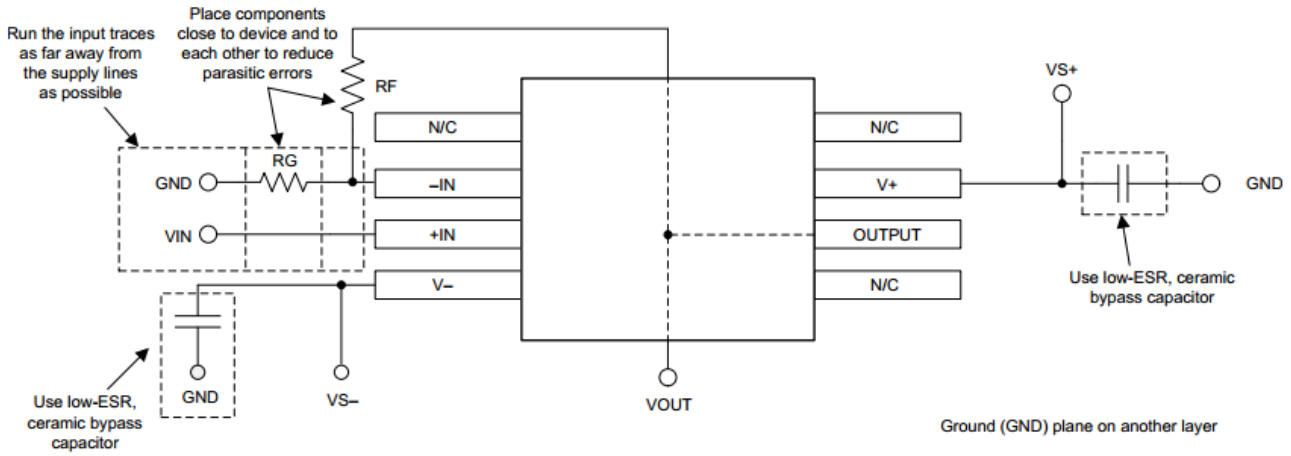


Figure 39. Operational Amplifier Board Layout for Noninverting Configuration

11.3 Power Dissipation

Power dissipation depends on power-supply voltage, signal and load conditions. With DC signals, power dissipation is equal to the product of output current times the voltage across the conducting output transistor, $V_S - V_O$. Power dissipation can be minimized by using the lowest possible power-supply voltage necessary to assure the required output voltage swing.

For resistive loads, the maximum power dissipation occurs at a DC output voltage of one-half the power-supply voltage. Dissipation with AC signals is lower. AB-039 *Power Amplifier Stress and Power Handling Limitations* explains how to calculate or measure power dissipation with unusual signals and loads, and can be found at www.ti.com.

Power Dissipation (continued)

Any tendency to activate the thermal protection circuit indicates excessive power dissipation or an inadequate heatsink. For reliable operation, junction temperature must be limited to 150°C, maximum. To estimate the margin of safety in a complete design, increase the ambient temperature until the thermal protection is triggered at 160°C. The thermal protection should trigger more than 35°C above the maximum expected ambient condition of the application.

11.4 PowerPAD Thermally-Enhanced Package

In addition to the regular 5-pin SOT-23 and 9-pin VSSOP packages, the single and dual versions of the OPA354 also come in an 8-pin SOIC PowerPAD package. The 98-pin SO with PowerPAD is a standard size 8-pin SOIC package where the exposed leadframe on the bottom of the package can be soldered directly to the PCB to create an extremely low thermal resistance. This direct attachment enhances the OPA354 power dissipation capability significantly, and eliminates the use of bulky heatsinks and slugs traditionally used in thermal packages. This package can be easily mounted using standard PCB assembly techniques.

NOTE

Because the 8-pin HSOP PowerPAD is pin-compatible with standard 8-pin SOIC packages, the OPA354 and OPA2354 can directly replace operational amplifiers in existing sockets. Soldering the PowerPAD to the PCB is always required, even with applications that have low power dissipation. This configuration provides the necessary thermal and mechanical connection between the leadframe die pad and the PCB.

The PowerPAD package is designed so that the leadframe die pad (or thermal pad) is exposed on the bottom of the IC, as shown in Figure 40. This exposed die provides an extremely low thermal resistance ($R_{\theta JC}$) path between the die and the exterior of the package. The thermal pad on the bottom of the IC can then be soldered directly to the PCB, using the PCB as a heatsink. In addition, plated-through holes (vias) provide a low thermal resistance heat flow path to the back side of the PCB.

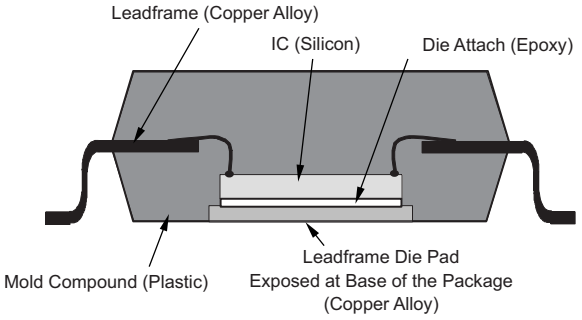


Figure 40. Section View of a PowerPAD Package

11.5 PowerPAD Assembly Process

The PowerPAD must be connected to the most negative supply voltage for the device, which is ground in single-supply applications and V− in split-supply applications.

Prepare the PCB with a top-side etch pattern, as shown in Figure 41. The exact land design may vary based on the specific assembly process requirements. There must be etch for the leads as well as etch for the thermal land.

Place the recommended number of plated-through holes (or thermal vias) in the area of the thermal pad. These holes must be 13 mils (.013 in) in diameter. They are kept small so that solder wicking through the holes is not a problem during reflow. TI recommends a minimum of 5 holes for the 8-pin HSOP PowerPAD package, as shown in Figure 41.

PowerPAD Assembly Process (continued)

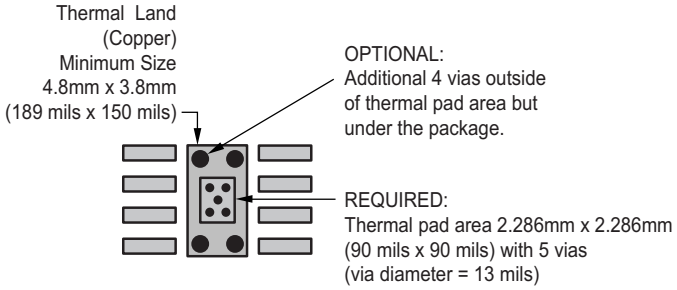


Figure 41. 8-Pin PowerPAD PCB Etch and Via Pattern

TI recommends, but does not require, placing a small number of additional holes under the package and outside the thermal pad area. These holes provide additional heat paths between the copper thermal land and the ground plane. They may be larger because they are not in the area to be soldered, so wicking is not a problem. This technique is illustrated in Figure 41.

Connect all holes, including those within the thermal pad area and outside the pad area, to the internal ground plane or other internal copper plane for single-supply applications, and to V− for split-supply applications.

When laying out these holes, do not use the typical web or spoke via connection methodology, as shown in Figure 42. Web connections have a high thermal resistance connection that is useful for slowing the heat transfer during soldering operations. This feature makes soldering the vias that have ground plane connections easier. However, in this application, low thermal resistance is desired for the most efficient heat transfer. Therefore, the holes under the PowerPAD package must make connection to the internal ground plane with a complete connection around the entire circumference of the plated-through hole.

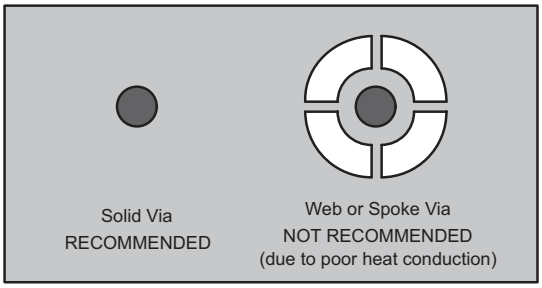


Figure 42. Via Connection

The top-side solder mask must leave the pad connections and the thermal pad area exposed. The thermal pad area must leave the 13-mil holes exposed. The larger holes outside the thermal pad area may be covered with solder mask.

Apply solder paste to the exposed thermal pad area and all of the package terminals.

With these preparatory steps in place, the PowerPAD IC is simply placed in position and run through the solder reflow operation as any standard surface-mount component. This preparation and processing results in a part that is properly installed.

For detailed information on the PowerPAD package including thermal modeling considerations and repair procedures, please see *PowerPAD Thermally Enhanced Package* located at www.ti.com.

12 Device and Documentation Support

12.1 Documentation Support

For related documentation see the following:

- *ADS8326 16-Bit, High-Speed, 2.7V to 5.5V microPower Sampling ANALOG-TO-DIGITAL CONVERTER* (SBAS343)
- *Circuit Board Layout Techniques* (SLOA089)
- *Compensate Transimpedance Amplifiers Intuitively* (SBOA055)
- *FilterPro™ User's Guide* (SBFA001)
- *Noise Analysis for High-Speed Op Amps*
- *OPA380 and OPA2380 Precision, High-Speed Transimpedance Amplifier* (SBOS291)
- *OPA355, OPA2355, and OPA3355 200MHz, CMOS OPERATIONAL AMPLIFIER WITH SHUTDOWN* (SBOS195)
- *OPA656 Wideband, Unity-Gain Stable, FET-Input OPERATIONAL AMPLIFIER* (SBOS196)
- *POWER AMPLIFIER STRESS AND POWER HANDLING LIMITATIONS* (SBOA022)
- *PowerPAD Thermally Enhanced Package* (SLMA002)

12.2 Related Links

Table 2 lists quick access links. Categories include technical documents, support and community resources, tools and software, and quick access to sample or buy.

Table 2. Related Links

PARTS	PRODUCT FOLDER	SAMPLE & BUY	TECHNICAL DOCUMENTS	TOOLS & SOFTWARE	SUPPORT & COMMUNITY
OPA354	Click here	Click here	Click here	Click here	Click here
OPA2354	Click here	Click here	Click here	Click here	Click here
OPA4354	Click here	Click here	Click here	Click here	Click here

12.3 Receiving Notification of Documentation Updates

To receive notification of documentation updates, navigate to the device product folder on ti.com. In the upper right corner, click on *Alert me* to register and receive a weekly digest of any product information that has changed. For change details, review the revision history included in any revised document.

12.4 Community Resources

The following links connect to TI community resources. Linked contents are provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

TI E2E™ Online Community ***TI's Engineer-to-Engineer (E2E) Community.*** Created to foster collaboration among engineers. At e2e.ti.com, you can ask questions, share knowledge, explore ideas and help solve problems with fellow engineers.

Design Support ***TI's Design Support*** Quickly find helpful E2E forums along with design support tools and contact information for technical support.

12.5 Trademarks

PowerPAD, E2E are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

12.6 Electrostatic Discharge Caution

These devices have limited built-in ESD protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

12.7 Glossary

SLYZ022 — *TI Glossary*.

This glossary lists and explains terms, acronyms, and definitions.

13 Mechanical, Packaging, and Orderable Information

The following pages include mechanical, packaging, and orderable information. This information is the most current data available for the designated devices. This data is subject to change without notice and revision of this document. For browser-based versions of this data sheet, refer to the left-hand navigation.

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
OPA2354AIDDA	ACTIVE	SO PowerPAD	DDA	8	75	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125	OPA 2354A	Samples
OPA2354AIDDAG3	ACTIVE	SO PowerPAD	DDA	8	75	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125	OPA 2354A	Samples
OPA2354AIDDAR	ACTIVE	SO PowerPAD	DDA	8	2500	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125	OPA 2354A	Samples
OPA2354AIDDARG3	ACTIVE	SO PowerPAD	DDA	8	2500	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125	OPA 2354A	Samples
OPA2354AIDGKR	ACTIVE	VSSOP	DGK	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAUAG	Level-2-260C-1 YEAR	-40 to 125	OACI	Samples
OPA2354AIDGKRG4	ACTIVE	VSSOP	DGK	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAUAG	Level-2-260C-1 YEAR	-40 to 125	OACI	Samples
OPA2354AIDGKT	ACTIVE	VSSOP	DGK	8	250	Green (RoHS & no Sb/Br)	CU NIPDAUAG	Level-2-260C-1 YEAR	-40 to 125	OACI	Samples
OPA2354AIDGKTG4	ACTIVE	VSSOP	DGK	8	250	Green (RoHS & no Sb/Br)	CU NIPDAUAG	Level-2-260C-1 YEAR	-40 to 125	OACI	Samples
OPA354AIDBVR	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OABI	Samples
OPA354AIDBVRG4	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OABI	Samples
OPA354AIDBVT	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OABI	Samples
OPA354AIDBVTG4	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OABI	Samples
OPA354AIDDA	ACTIVE	SO PowerPAD	DDA	8	75	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125	OPA 354A	Samples
OPA354AIDDAG3	ACTIVE	SO PowerPAD	DDA	8	75	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125	OPA 354A	Samples
OPA354AIDDAR	ACTIVE	SO PowerPAD	DDA	8	2500	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125	OPA 354A	Samples
OPA4354AID	ACTIVE	SOIC	D	14	50	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OPA4354A	Samples
OPA4354AIDG4	ACTIVE	SOIC	D	14	50	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OPA4354A	Samples

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
OPA4354AIDR	ACTIVE	SOIC	D	14	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OPA4354A	Samples
OPA4354AIDRG4	ACTIVE	SOIC	D	14	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OPA4354A	Samples
OPA4354AIPWR	ACTIVE	TSSOP	PW	14	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OPA 4354A	Samples
OPA4354AIPWRG4	ACTIVE	TSSOP	PW	14	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OPA 4354A	Samples
OPA4354AIPWT	ACTIVE	TSSOP	PW	14	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OPA 4354A	Samples
OPA4354AIPWTG4	ACTIVE	TSSOP	PW	14	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR	-40 to 125	OPA 4354A	Samples

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. - The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

(4) There may be additional marking, which relates to the logo, the lot trace code information, or the environmental category on the device.

(5) Multiple Device Markings will be inside parentheses. Only one Device Marking contained in parentheses and separated by a "~" will appear on a device. If a line is indented then it is a continuation of the previous line and the two combined represent the entire Device Marking for that device.

(6) Lead/Ball Finish - Orderable Devices may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead/Ball Finish values may wrap to two lines if the finish value exceeds the maximum column width.

OTHER QUALIFIED VERSIONS OF OPA4354 :

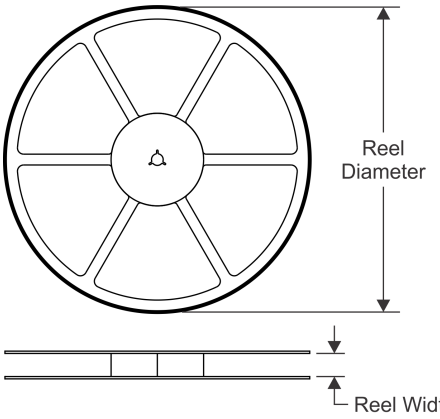
- Automotive: OPA4354-Q1

NOTE: Qualified Version Definitions:

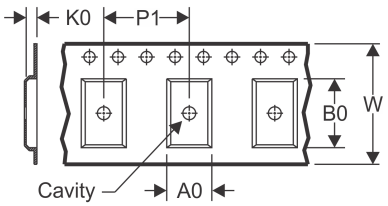
- Automotive - Q100 devices qualified for high-reliability automotive applications targeting zero defects

TAPE AND REEL INFORMATION

REEL DIMENSIONS

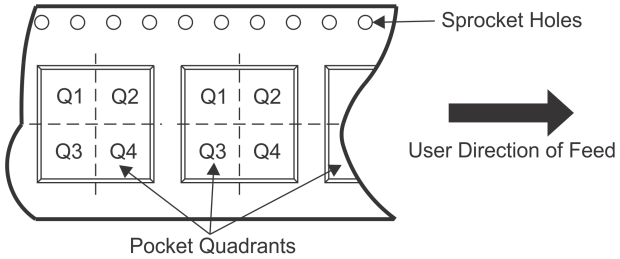


TAPE DIMENSIONS



A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers

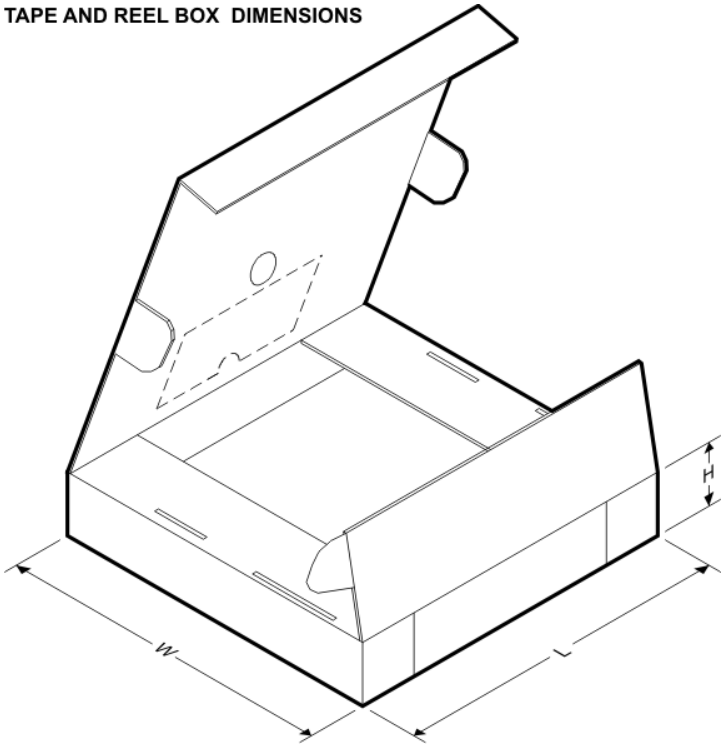
QUADRANT ASSIGNMENTS FOR PIN 1 ORIENTATION IN TAPE



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
OPA2354AIDDAR	SO Power PAD	DDA	8	2500	330.0	12.4	6.4	5.2	2.1	8.0	12.0	Q1
OPA2354AIDGKR	VSSOP	DGK	8	2500	330.0	12.4	5.3	3.4	1.4	8.0	12.0	Q1
OPA2354AIDGKT	VSSOP	DGK	8	250	330.0	12.4	5.3	3.4	1.4	8.0	12.0	Q1
OPA354AIDBVR	SOT-23	DBV	5	3000	178.0	9.0	3.23	3.17	1.37	4.0	8.0	Q3
OPA354AIDBVT	SOT-23	DBV	5	250	178.0	9.0	3.3	3.2	1.4	4.0	8.0	Q3
OPA354AIDDAR	SO Power PAD	DDA	8	2500	330.0	12.4	6.4	5.2	2.1	8.0	12.0	Q1
OPA4354AIDR	SOIC	D	14	2500	330.0	16.4	6.5	9.0	2.1	8.0	16.0	Q1
OPA4354AIPWR	TSSOP	PW	14	2500	330.0	12.4	6.9	5.6	1.6	8.0	12.0	Q1
OPA4354AIPWT	TSSOP	PW	14	250	180.0	12.4	6.9	5.6	1.6	8.0	12.0	Q1

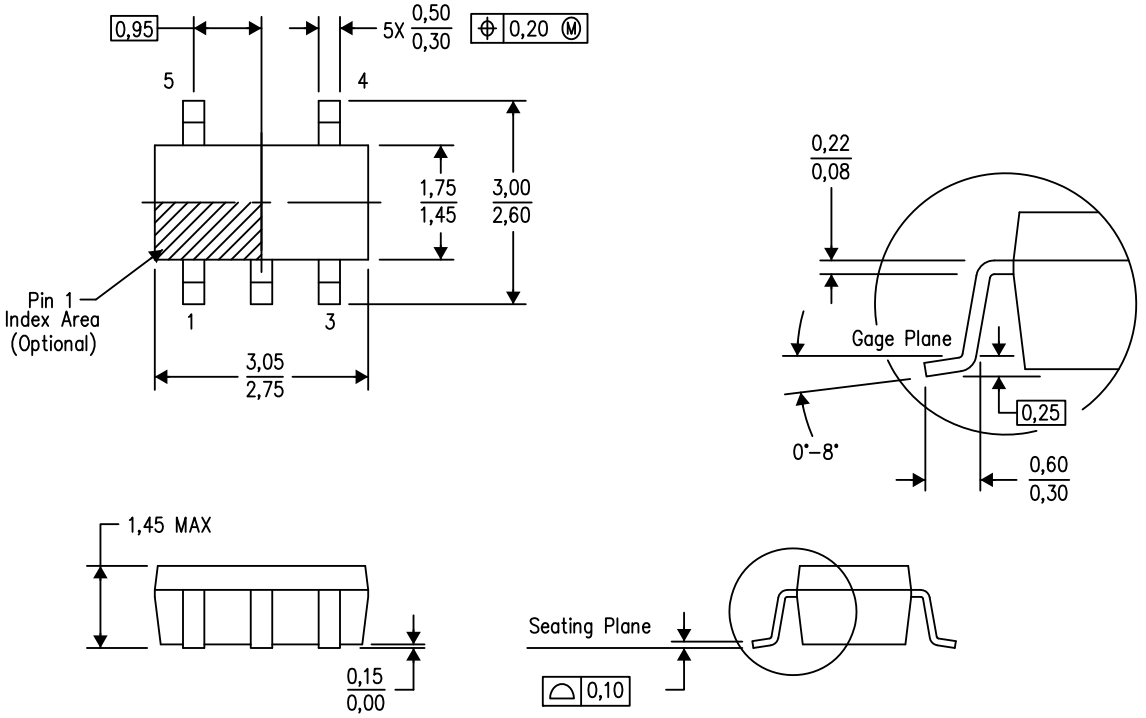
TAPE AND REEL BOX DIMENSIONS



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
OPA2354AIDDAR	SO PowerPAD	DDA	8	2500	367.0	367.0	35.0
OPA2354AIDGKR	VSSOP	DGK	8	2500	366.0	364.0	50.0
OPA2354AIDGKT	VSSOP	DGK	8	250	366.0	364.0	50.0
OPA354AIDBVR	SOT-23	DBV	5	3000	180.0	180.0	18.0
OPA354AIDBVT	SOT-23	DBV	5	250	180.0	180.0	18.0
OPA354AIDDAR	SO PowerPAD	DDA	8	2500	367.0	367.0	35.0
OPA4354AIDR	SOIC	D	14	2500	367.0	367.0	38.0
OPA4354AIPWR	TSSOP	PW	14	2500	367.0	367.0	35.0
OPA4354AIPWT	TSSOP	PW	14	250	210.0	185.0	35.0

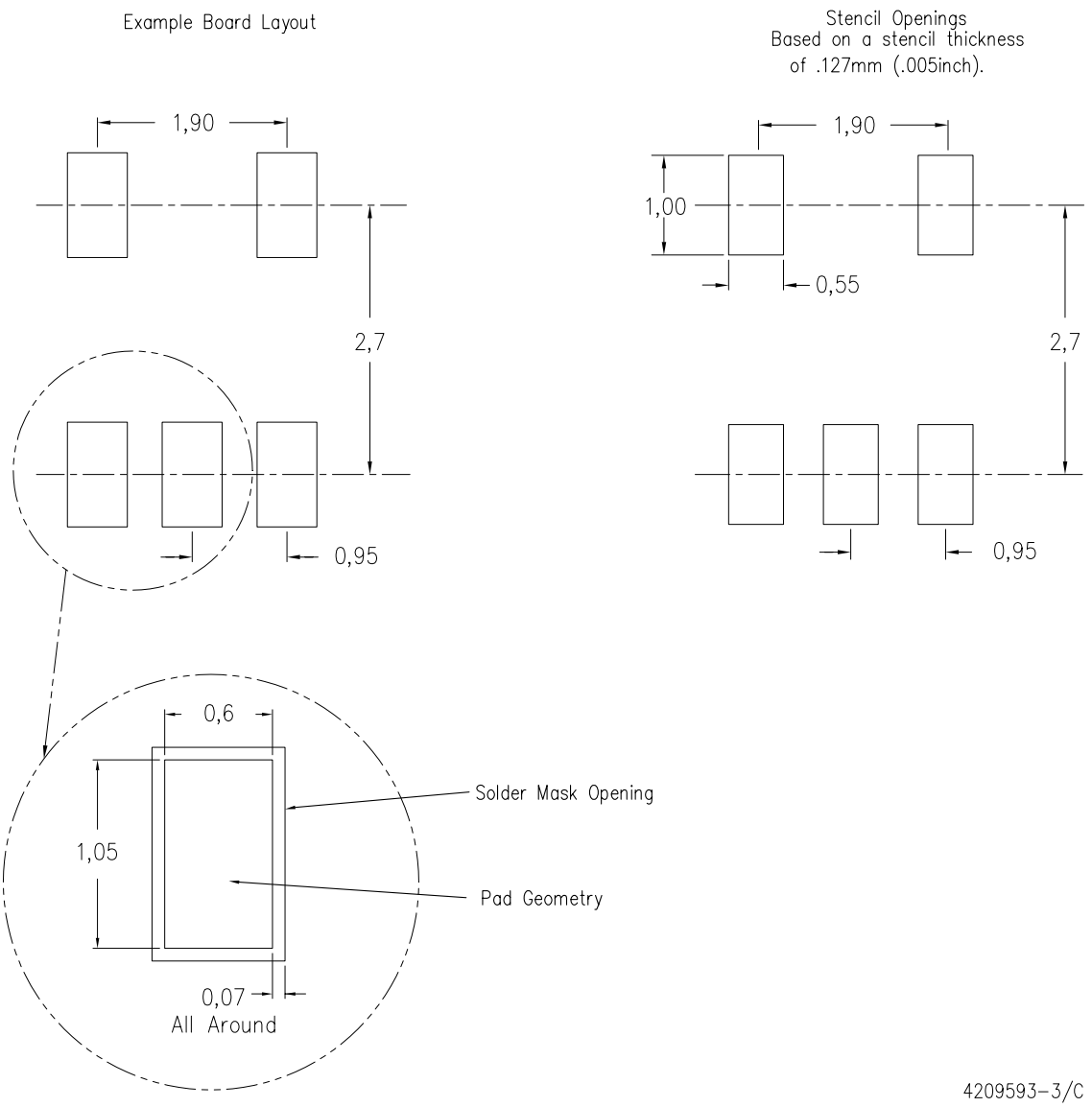
DBV (R–PDSO–G5) PLASTIC SMALL–OUTLINE PACKAGE



4073253–4/N 12/14

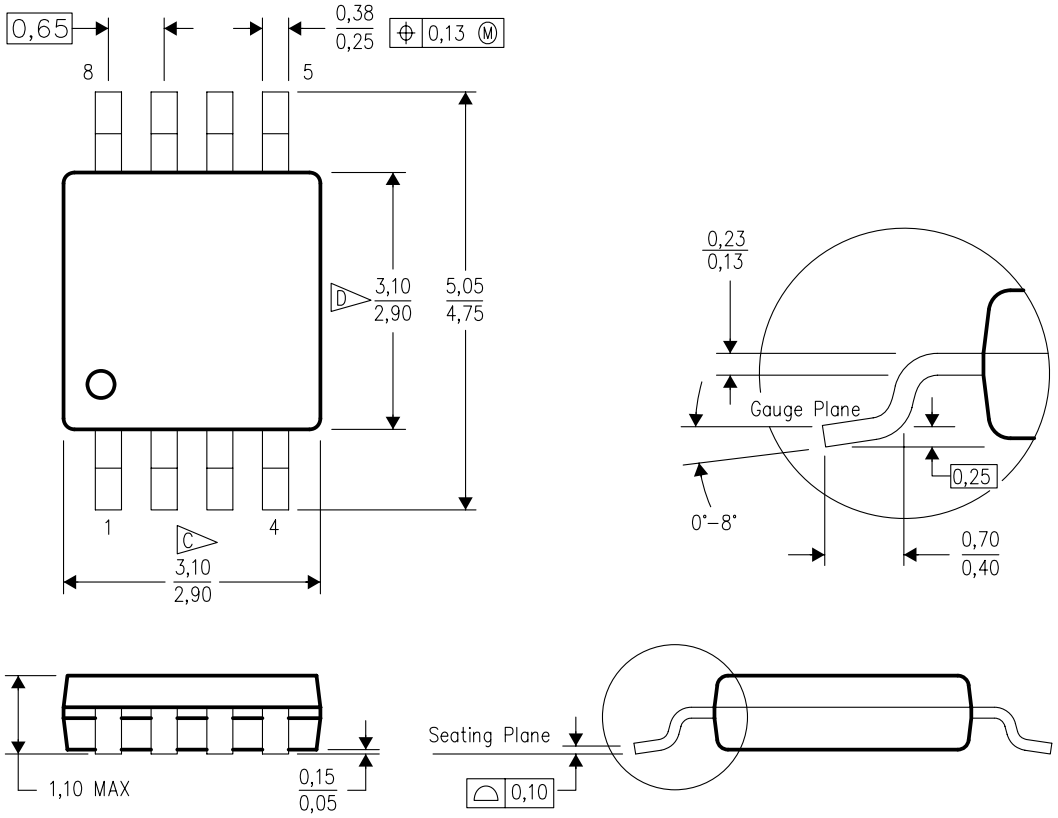
NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion. Mold flash and protrusion shall not exceed 0.15 per side.
D. Falls within JEDEC MO–178 Variation AA.

DBV (R–PDSO–G5) PLASTIC SMALL OUTLINE



NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Customers should place a note on the circuit board fabrication drawing not to alter the center solder mask defined pad.
D. Publication IPC–7351 is recommended for alternate designs.
E. Laser cutting apertures with trapezoidal walls and also rounding corners will offer better paste release. Customers should contact their board assembly site for stencil design recommendations. Example stencil design based on a 50% volumetric metal load solder paste. Refer to IPC–7525 for other stencil recommendations.

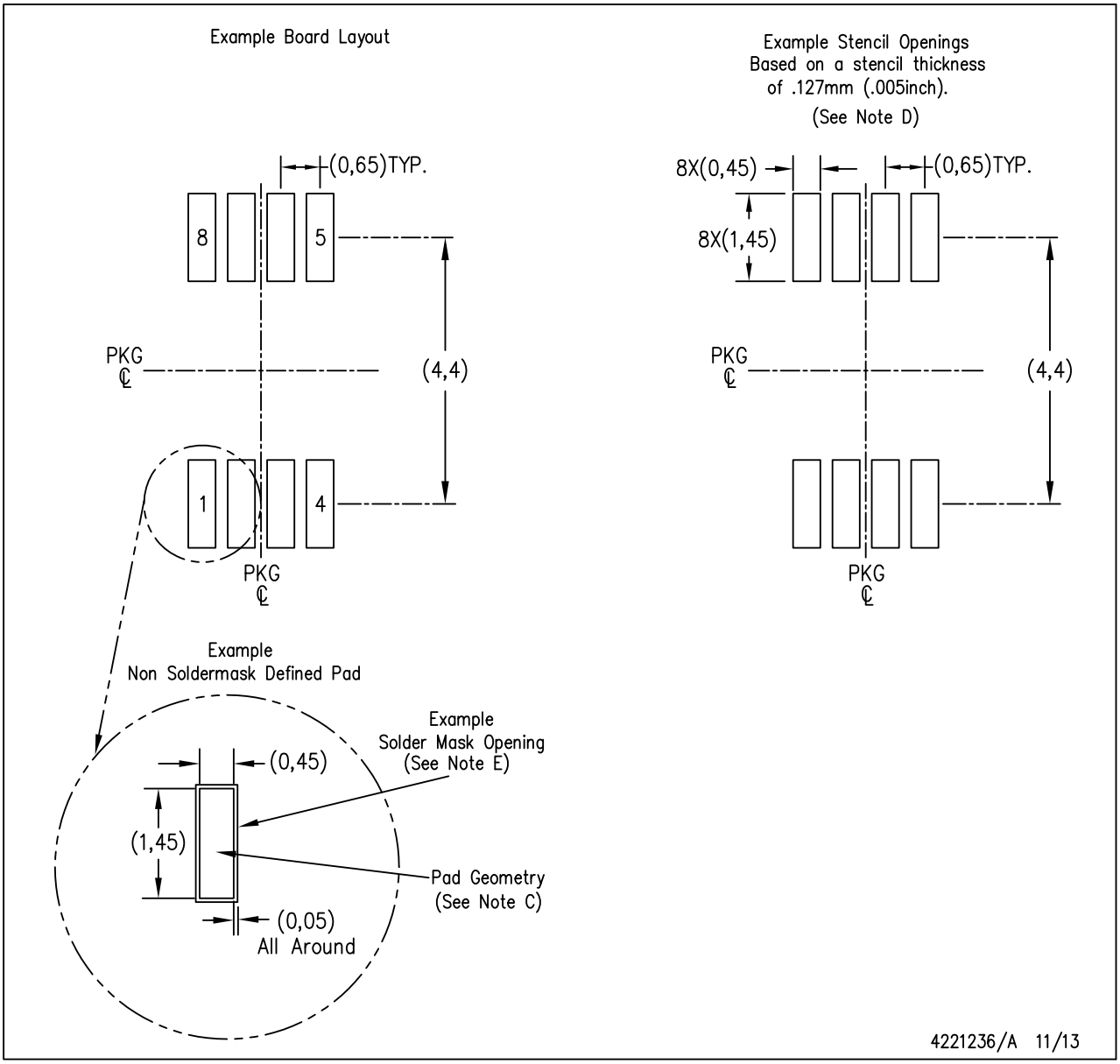
DGK (S–PDSO–G8) PLASTIC SMALL–OUTLINE PACKAGE



4073329/E 05/06

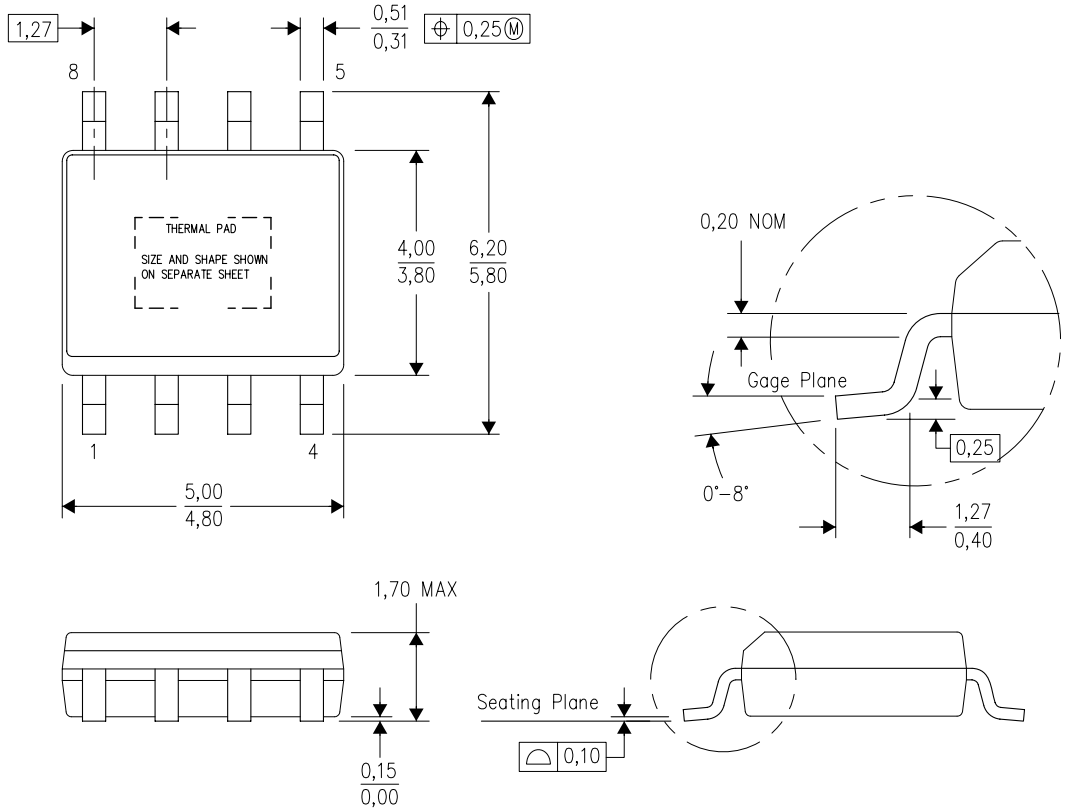
NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body length does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed 0.15 per end.
D. Body width does not include interlead flash. Interlead flash shall not exceed 0.50 per side.
E. Falls within JEDEC MO–187 variation AA, except interlead flash.

DGK (S–PDSO–G8) PLASTIC SMALL OUTLINE PACKAGE



NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Publication IPC–7351 is recommended for alternate designs.
D. Laser cutting apertures with trapezoidal walls and also rounding corners will offer better paste release. Customers should contact their board assembly site for stencil design recommendations. Refer to IPC–7525 for other stencil recommendations.
E. Customers should contact their board fabrication site for solder mask tolerances between and around signal pads.

DDA (R–PDSO–G8) PowerPAD™ PLASTIC SMALL–OUTLINE



4202561/F 12/11

NOTES:
A. All linear dimensions are in millimeters. Dimensioning and tolerancing per ASME Y14.5–1994.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion not to exceed 0,15.
D. This package is designed to be soldered to a thermal pad on the board. Refer to Technical Brief, PowerPad Thermally Enhanced Package, Texas Instruments Literature No. SLMA002 for information regarding recommended board layout. This document is available at www.ti.com <http://www.ti.com>.
E. See the additional figure in the Product Data Sheet for details regarding the exposed thermal pad features and dimensions.
F. This package complies to JEDEC MS–012 variation BA

PowerPAD is a trademark of Texas Instruments.

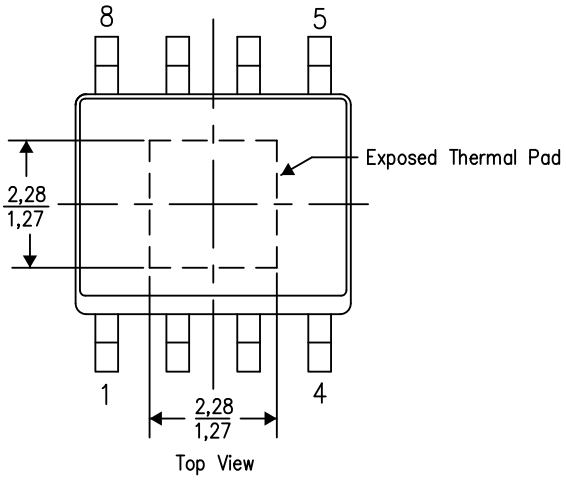
DDA (R–PDSO–G8) PowerPAD™ PLASTIC SMALL OUTLINE

THERMAL INFORMATION

This PowerPAD™ package incorporates an exposed thermal pad that is designed to be attached to a printed circuit board (PCB). The thermal pad must be soldered directly to the PCB. After soldering, the PCB can be used as a heatsink. In addition, through the use of thermal vias, the thermal pad can be attached directly to the appropriate copper plane shown in the electrical schematic for the device, or alternatively, can be attached to a special heatsink structure designed into the PCB. This design optimizes the heat transfer from the integrated circuit (IC).

For additional information on the PowerPAD package and how to take advantage of its heat dissipating abilities, refer to Technical Brief, PowerPAD Thermally Enhanced Package, Texas Instruments Literature No. SLMA002 and Application Brief, PowerPAD Made Easy, Texas Instruments Literature No. SLMA004. Both documents are available at www.ti.com.

The exposed thermal pad dimensions for this package are shown in the following illustration.



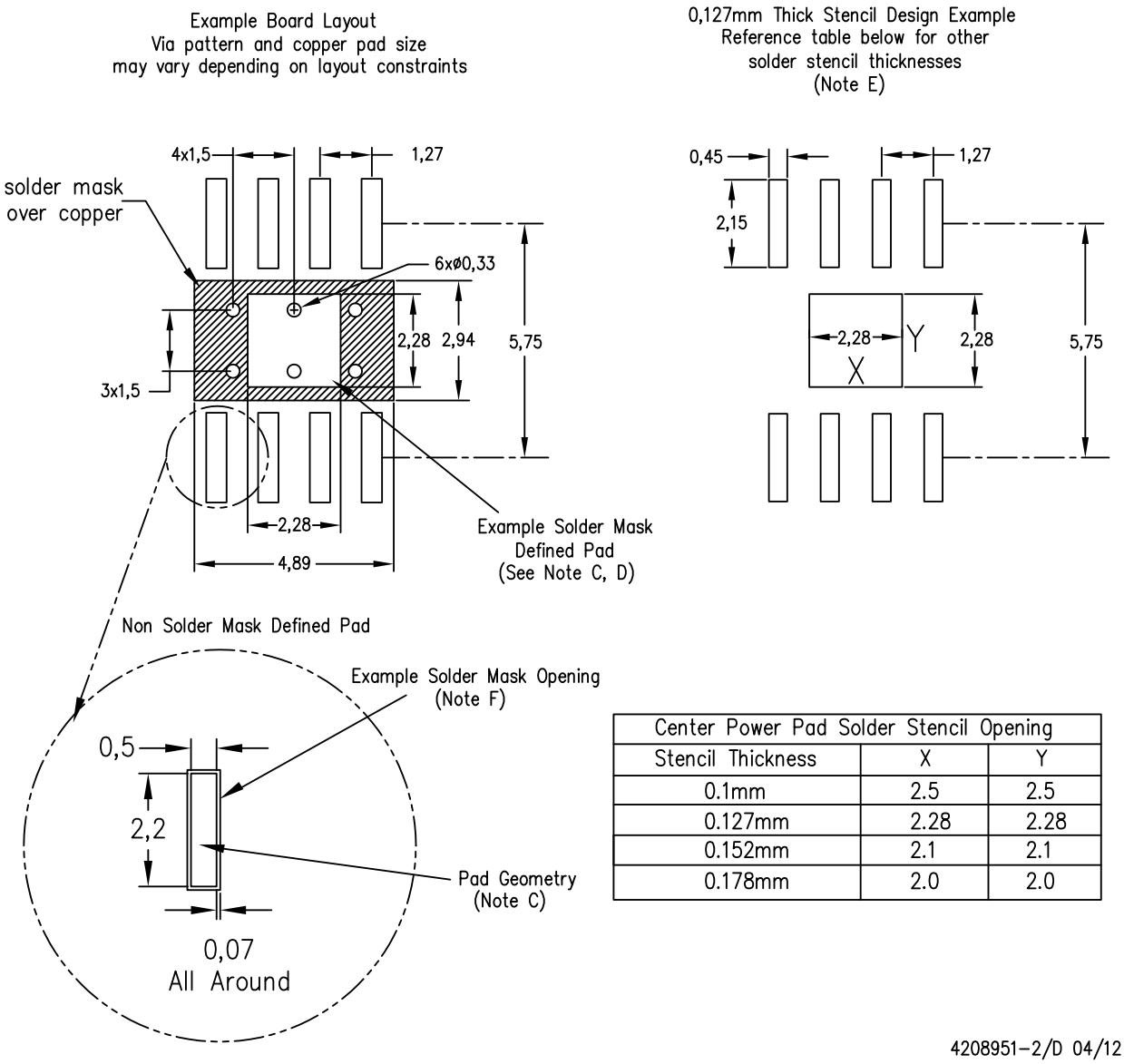
Exposed Thermal Pad Dimensions

4206322–2/L 05/12

NOTE: A. All linear dimensions are in millimeters

PowerPAD is a trademark of Texas Instruments

DDA (R–PDSO–G8) PowerPAD™ PLASTIC SMALL OUTLINE



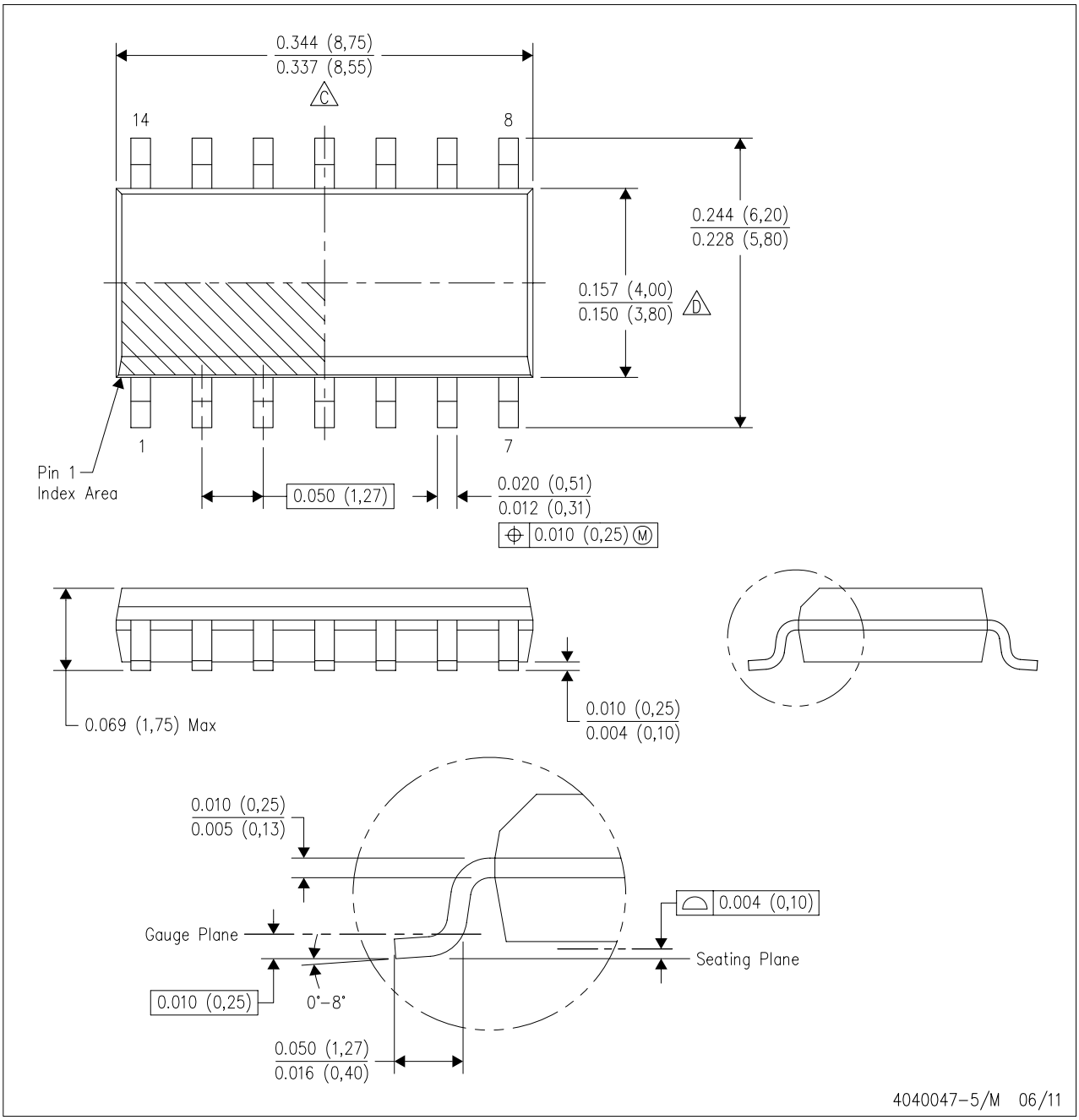
Center Power Pad Solder Stencil Opening		
Stencil Thickness	X	Y
0.1mm	2.5	2.5
0.127mm	2.28	2.28
0.152mm	2.1	2.1
0.178mm	2.0	2.0

4208951–2/D 04/12

NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Publication IPC–7351 is recommended for alternate designs.
D. This package is designed to be soldered to a thermal pad on the board. Refer to Technical Brief, PowerPad Thermally Enhanced Package, Texas Instruments Literature No. SLMA002, SLMA004, and also the Product Data Sheets for specific thermal information, via requirements, and recommended board layout. These documents are available at www.ti.com <http://www.ti.com>. Publication IPC–7351 is recommended for alternate designs.
E. Laser cutting apertures with trapezoidal walls and also rounding corners will offer better paste release. Customers should contact their board assembly site for stencil design recommendations. Example stencil design based on a 50% volumetric metal load solder paste. Refer to IPC–7525 for other stencil recommendations.
F. Customers should contact their board fabrication site for solder mask tolerances between and around signal pads.

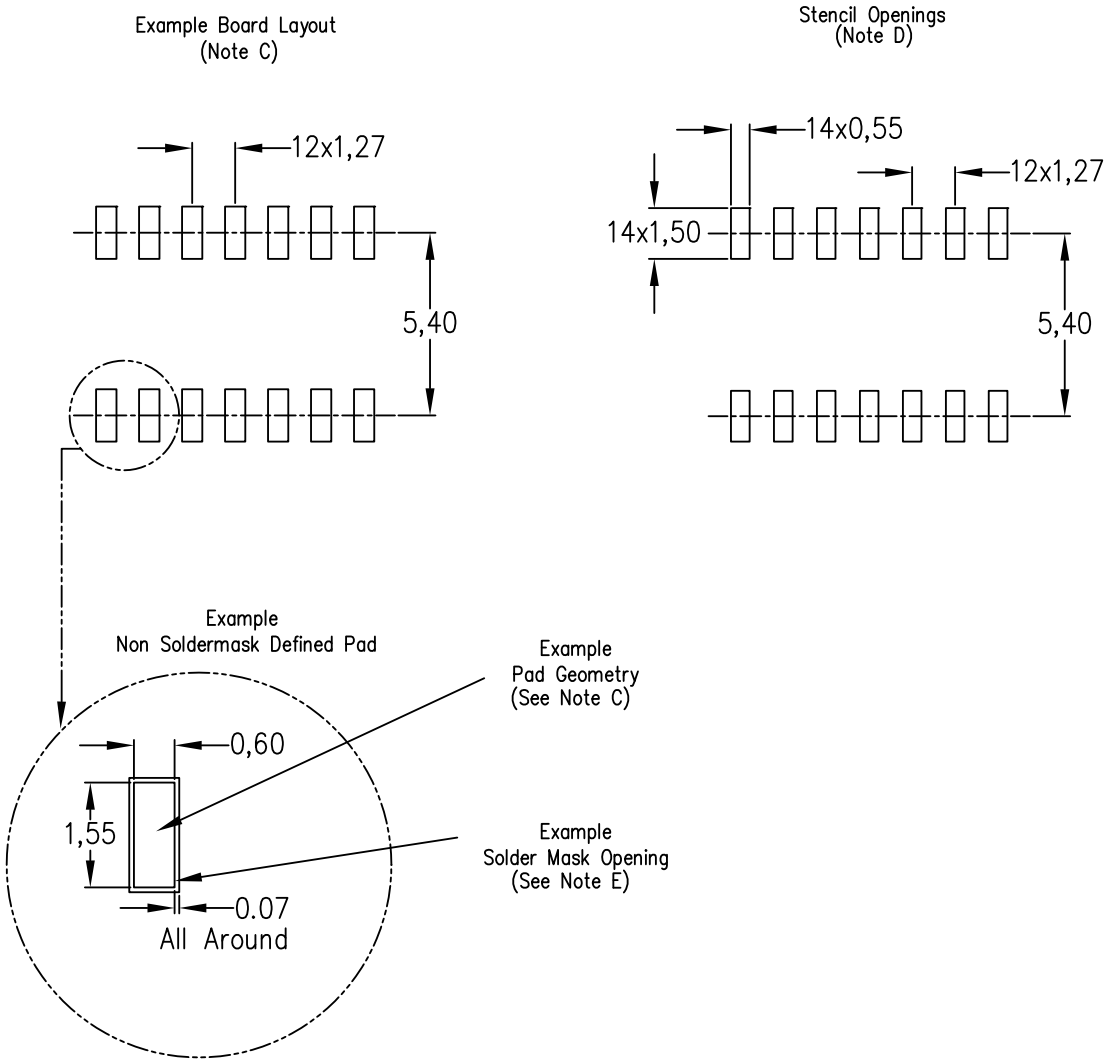
PowerPAD is a trademark of Texas Instruments.

D (R–PDSO–G14) PLASTIC SMALL OUTLINE



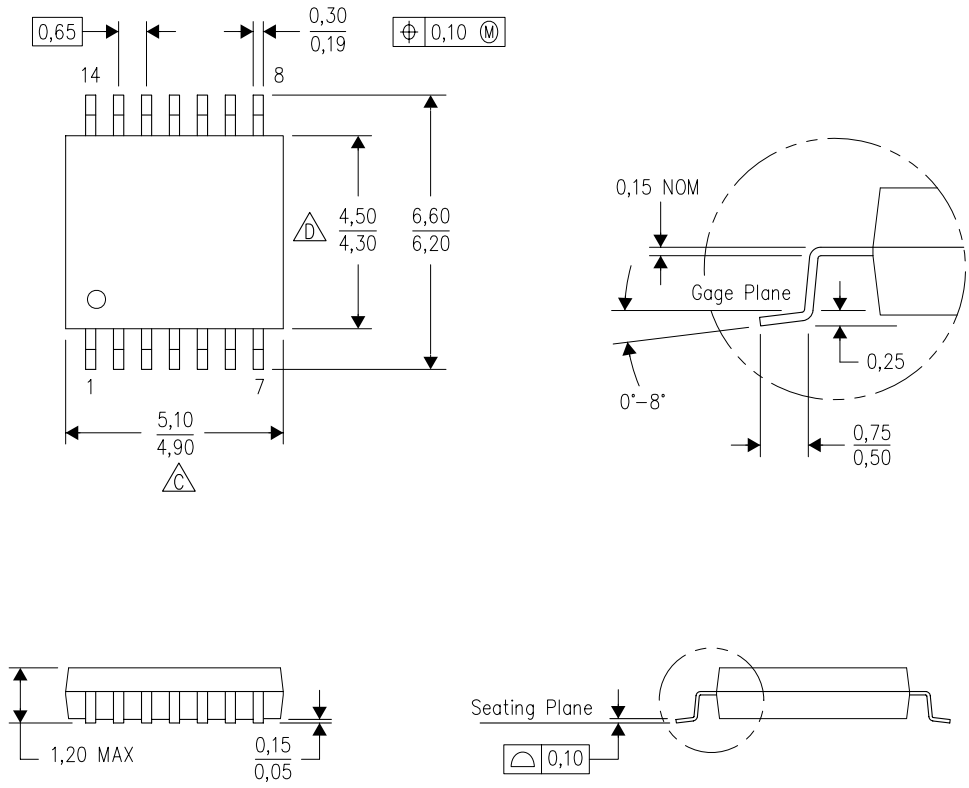
NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Body length does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed 0.006 (0,15) each side.
D. Body width does not include interlead flash. Interlead flash shall not exceed 0.017 (0,43) each side.
E. Reference JEDEC MS–012 variation AB.

D (R–PDSO–G14) PLASTIC SMALL OUTLINE



NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Publication IPC–7351 is recommended for alternate designs.
D. Laser cutting apertures with trapezoidal walls and also rounding corners will offer better paste release. Customers should contact their board assembly site for stencil design recommendations. Refer to IPC–7525 for other stencil recommendations.
E. Customers should contact their board fabrication site for solder mask tolerances between and around signal pads.

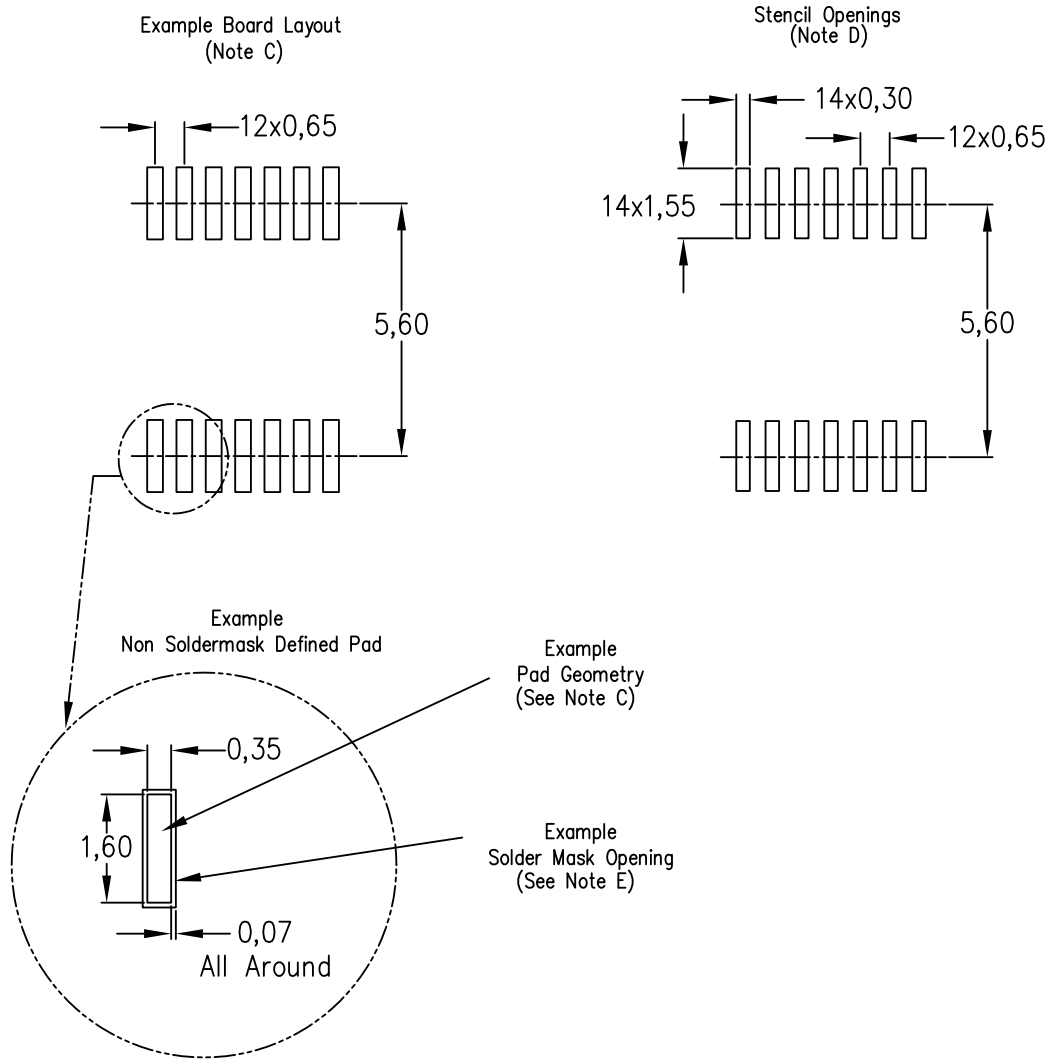
PW (R–PDSO–G14) PLASTIC SMALL OUTLINE



4040064–3/G 02/11

NOTES:
A. All linear dimensions are in millimeters. Dimensioning and tolerancing per ASME Y14.5M–1994.
B. This drawing is subject to change without notice.
C. Body length does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed 0,15 each side.
D. Body width does not include interlead flash. Interlead flash shall not exceed 0,25 each side.
E. Falls within JEDEC MO–153

PW (R–PDSO–G14) PLASTIC SMALL OUTLINE



NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Publication IPC–7351 is recommended for alternate designs.
D. Laser cutting apertures with trapezoidal walls and also rounding corners will offer better paste release. Customers should contact their board assembly site for stencil design recommendations. Refer to IPC–7525 for other stencil recommendations.
E. Customers should contact their board fabrication site for solder mask tolerances between and around signal pads.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

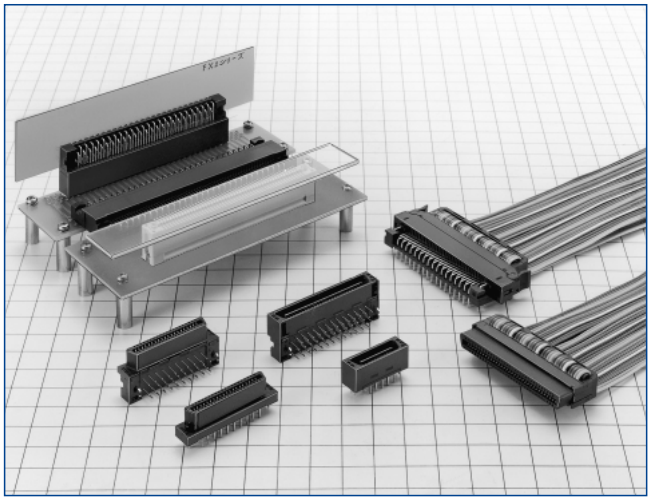
TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

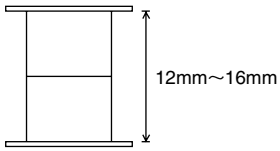
Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

1.27mm Pitch Multi-function Two Piece Connectors

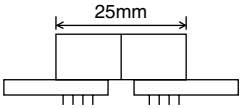
FX2 Series



Stacking connection (Stack height : 12～16mm)



Horizontal Connection



Vertical Connection

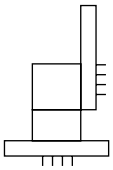
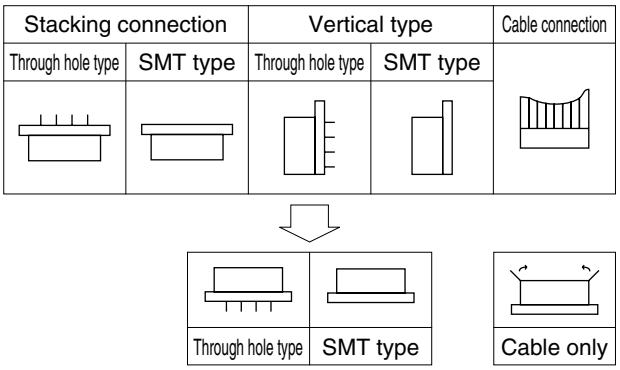
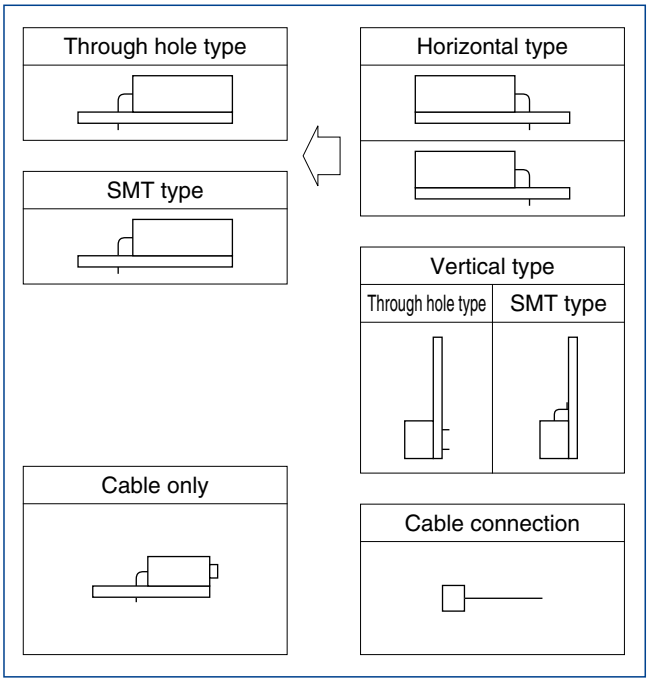


Fig.1

■Features

1. Various connection with various product line





2. Easy One-Touch Operation

The ribbon cable connection type allows easy one-touch operation with either single hand.

Insersion and Extraction

(1) Pick internal locks with thumb and the index finger.

Fig.2

(2) With unique and preferable click feeling, the cable and connector can be inserted or withdrawed.

Fig.3

(For insertion, the operation proceeds from procedure (2) to (1).)

3. Board to board type

① Prevents flux penetration

The straight through hole type takes the complete flux tight action from the board back side at solder dipping. The type is prepared, corresponding to whether or not cleaning is required.

② Stack height 12mm to 16mm

The board stack height can be set to every 1 mm step up to 12 to 16mm.

③ Mis-insertion preventive mechanism

The mating area is designed in a mechanism so as to prevent mis-insertion, and complete countermeasures have been taken against wrenching.

4. Board to cable type

① Positive lock with easy operation

Employing the inner lock system, positive lock and eject actions can be performed with easy operation.

② Applicable cable

The applicable cable is the <UL2651>28 AWG flat cable (7/0.127mm), and the jacket size is 0.9±0.1mm.

5. SMT type

Robust design with metal hold-down

The right angle type is equipped with metal hold-down to secure soldering strength, and constructed so as to fix by screws. The straight type can choose whether or not metal hold-down are required.

■Product Specifications

Rating	Current rating 0.5A Voltage rating 125V AC	Operating Temperature Range -55℃ to +85℃ (Note 1) Operating Humidity Range 40 to 80%	Storage Temperature Range -10℃ to +60℃(Note 2) Storage Humidity Range 40 to 70%(Note 2)

Item	Specification	Condition	
1. Insulation Resistance	1000MΩ min	250V DC	
2. Withstanding Voltage	No flashover or insulation breakdown.	300V AC/1 min	
3. Contact Resistance	45mΩ max.	100mA	
4. Vibration	No electrical discontinuity of 1μs or more	Frequency: 10 to 55 Hz, single amplitude of 0.75 mm, 2 hours in each of the 3 directions.	
5. Humidity (Steady state)	Contact resistance: 55mΩ max. Insulation resistance: 100MΩ min.	96 hours at temperature of 40℃ and humidity of 90% to 95%	
6. Temperature Cycle	Contact resistance: 55mΩ max. Insulation resistance: 100MΩ min. No damage, cracks, or parts looseness.	(-55℃: 30 minutes→15 to 35℃: 2 to 3 minutes→ 85℃: 30 minutes→15 to 35℃: 2 to 3 minutes) 5 cycles	
7. Durability (Mating/un-mating)	Contact resistance: 55mΩ max.	500 cycles	
8. Resistance to soldering heat	No deformation of components affecting performance.	SMT Type	Reflow: At the recommended temperature profile Manual soldering: 360℃ for 5 seconds
		DIP Type	Solder bath: 260℃ for 10 seconds Manual soldering: 360℃ for 5 seconds

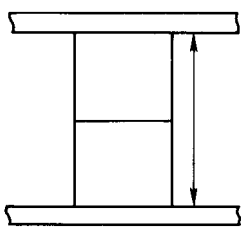
Note 1 : Includes temperature rise caused by current flow.
Note 2 : The term "storage" refers to products stored for long period of time prior to mounting and use. Operating Temperature Range and Humidity range covers non conducting condition of installed connectors in storage, shipment or during transportation.

■Material

Parts		Material	Finish	Remark
Insulator	Through hole type	Polyamid	Black	UL94V-0
	Socket	PBT/Polyamid		
	SMT type	PPS	Light brown	
Contact	Receptacle	Phosphor bronze	Selective gold plated	————
	Socket			
	Header	Phosphor bronze or brass		

■Stacking Variation

Unit : mm



Header \ Receptacle		Through hole type		SMT type
		FX2C-∗S-1.27DSA(L)	FX2C2-∗S-1.27DSA(L)	FX2-∗S-1.27SV(L)
		FX2CA-∗S-1.27DSA(L)	FX2CA2-∗S-1.27DSA(L)	
Through hole type	FX2C-∗P-1.27DSA(L)	12	14.0	12.2
	FX2CA-∗P-1.27DSA(L)			
	FX2CA1-∗P-1.27DSA(L)	13	15.0	13.2
	FX2CA2-∗P-1.27DSA(L)	14	16.0	14.2
SMT type	FX2-∗P-1.27SV(L)	12.1	14.1	12.3

■Product Number Structure

●Board to Board Receptacle

FX2 CA 2 - ＊ S - 1.27 DSA L
❶ ❷ ❸ ❹ ❺ ❻ ❼ ❽

❶Series Name : FX2	❹No. of contacts : 20, 32, 40, 52, 60, 68, 80, 100, 120
❷Blank : Right angle type C : Straight non-cleaning type CA : Straight cleaning type	❺Connector type : S : Receptacle
	❻Contact pitch : 1.27mm
❸Product height variation (DSA only) Blank : Standard product 2 : +2mm	❼Contact type : DS : Right angle type DSA : Straight type
	❽L : Board prefixed pin

●Board to Board Header

FX2 CA 1 - ＊ P - 1.27 DSA L
❶ ❷ ❸ ❹ ❺ ❻ ❼ ❽

❶Series Name : FX2	❹No. of contacts : 20, 32, 40, 52, 60, 68, 80, 100, 120
❷Blank : Right angle type C : Straight non-cleaning type CA : Straight cleaning type	❺Connector type : P : header
	❻Contact pitch : 1.27mm
❸Product height variation (DSA only) Blank : Standard product 1 : +1mm 2 : +2mm	❼Contact type : DS : Right angle type DSA : Straight type
	❽L : Board prefixed pin

Note 1 : C and CA take complete countermeasures against flux.
Note 2 : Cleaning type: the liquid escape hole is added at dip cleaning.
Note 3 : No-cleaning type: no liquid escape hole is added at dip cleaning.

●Board to Cable Socket

FX2 BA - ＊ S A - 1.27 R
❶ ❷ ❸ ❹ ❺ ❻ ❼

❶Series Name : FX2	❹S : Socket
❷B : Lock cable type BA : No lock cable type	❺A : Contact material: phosphor bronze
	❻Contact pitch : 1.27mm
❸No. of contacts : 20, 32, 40, 52, 60, 68, 80, 100	❼R : Insulation displacement

●Board to Cable Header

FX2 BA - ＊ P A - 1.27 DSA L
❶ ❷ ❸ ❹ ❺ ❻ ❼ ❽

❶Series Name : FX2	❺None : Contact material: phosphor bronze A : Contact material: brass
❷B : Lock straight no cleaning type : Lock right angle type BA : Lock straight cleaning type	❻Contact pitch : 1.27mm
❸No. of contacts : 20, 32, 40, 52, 60, 68, 80, 100	❼Contact type : DS : Right angle type DSA : Straight type
❹Connector type : P: header	❽L : Board prefixed pin

Note 1 : Cleaning type: The liquid escape hole is added at dip cleaning.
Note 2 : No cleaning type: No liquid escape hole is added at dip cleaning.
Note 3 : Straight 20 and 32 contact types use phosphor bronze for contact material.

●Straight SMT Type

FX2 - ✻ P - 1.27 SV L
❶ ❷ ❸ ❹ ❺ ❻

❶Series Name : FX2	❹Contact pitch : 1.27mm
❷No. of contacts : 20, 32, 40, 52, 60, 68, 80, 100, 120	❺Contact type : SV : Straight SMT type
❸Connector type : P : Header S : Receptacle	❻L : Board prefixed pin

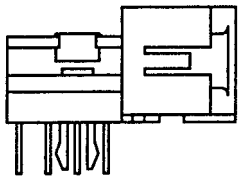
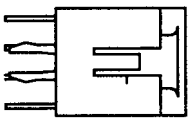
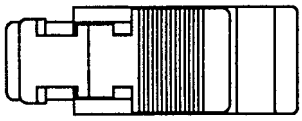
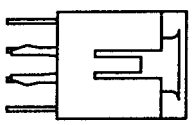
●Header Right Angle SMT Type

FX2 A - ✻ P - 0.635 SH
❶ ❷ ❸ ❹ ❺ ❻

❶Series Name : FX2	❹Connector type : P : Header
❷Blank : With boss A : Without boss	❺Mounting area pitch : 0.635mm
❸No. of contacts : 20, 40, 52, 60, 80	❻Contact type : SH : Right angle SMT type

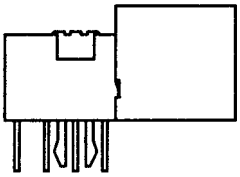
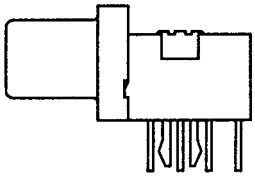
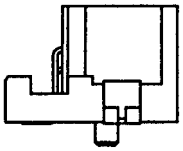
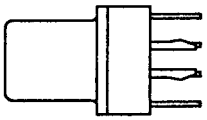
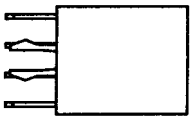
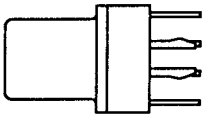
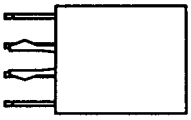
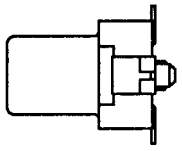
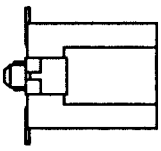
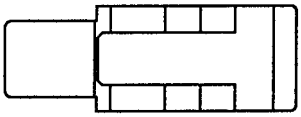
◆FX2 Functional Flow Chart

Board to Board Cable Type

Header Side	Receptacle and Socket Side
FX2B-✻PA-1.27DS (L) **Right angle type with lock** 	
FX2B-✻PA-1.27DSA(L) **Straight type with lock** **No cleaning type, flux tight** 	**FX2BA-✻SA-1.27R** **Cable type with lock** 
FX2BA-✻PA-1.27DSA(L) **Straight type with lock** **Cleaning type, flux tight** 	

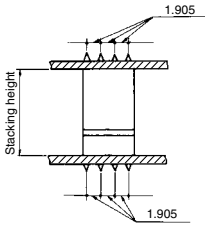
◆FX2 Functional Flow Chart

Board to Board Cable Type

Header Side	Receptacle and Socket Side
FX2-∗P-1.27DS (L) **Right angle type** 	**FX2-∗S-1.27DS (L)** **Right angle type** 
FX2(A)-∗P-0.635SH **Right angle type** **SMT type** 	**FX2C-∗S-1.27DS (L)** **Straight type** **No cleaning type, flux tight** 
FX2C-∗P-1.27DSA (L) **Straight type** **No cleaning type, flux tight** 	**FX2C-∗S-1.27DS (L)** **Straight type** **Cleaning type, flux tight** 
FX2CA-∗P-1.27DSA (L) **Straight type** **Cleaning type, flux tight** 	**FX2CA-∗S-1.27DSV (L)** **Straight type** **SMT type** 
FX2-∗P-1.27DSA (L) **Straight type** **SMT type** 	**FX2BA-∗SA-1.27R** **Cable type without lock** 

◆Through hole Type Application Pattern

Header



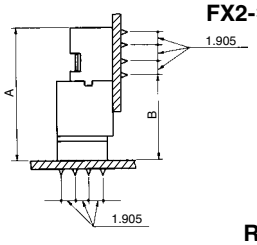
Stacking height

Header \ Receptacle	FX2CA-∗S	FX2CA2-∗S
FX2CA-∗P	12	14
FX2CA1-∗P	13	15
FX2CA2-∗P	14	16

Receptacle

FX2-∗P-1.27DS(L)

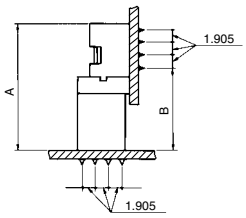
FX2-∗P-1.27DS(L)



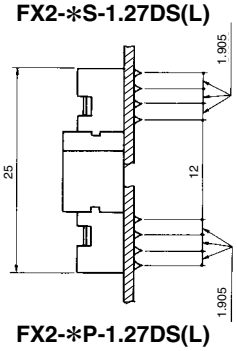
Receptacle	A	B
FX2C(A) -∗S	18.5	12
FX2C(A)2-∗S	20.5	14

Receptacle

FX2-∗S-1.27DS(L)



Header	A	B
FX2C(A) -∗P	18.5	12
FX2C(A)1-∗P	19.5	13
FX2C(A)2-∗P	20.5	14

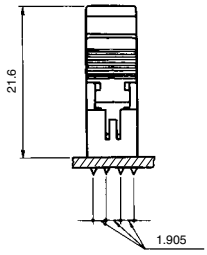


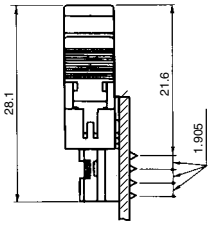
Header

FX2B(A)-∗SA-1.27R

FX2-∗P-1.27DS(L)

FX2B(A)-∗SA-1.27R



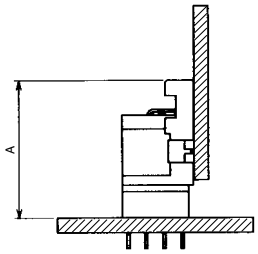


FX2B(A)-∗P(A)-1.27DSA(L)

FX2B-∗PA-1.27DS(L)

◆SMT Type Application Pattern

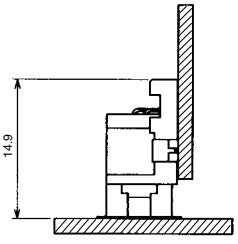
FX2(A)-∗P-0.635SH



Receptacle	A
FX2C(A)	14.7
FX2C(A)2	16.7

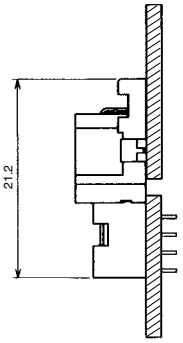
FX2C(A·2)-∗S-1.27DSA(L)

FX2(A)-∗P-0.635SH



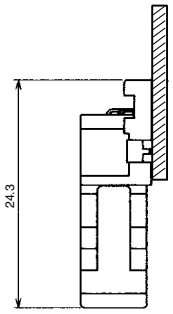
FX2-∗S-1.27SV(L)

FX2(A)-∗P-0.635SH



FX2-∗S-1.27DS(L)

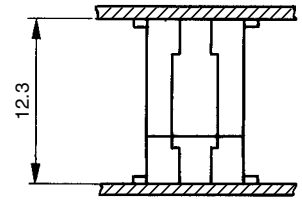
FX2(A)-∗P-0.635SH



FX2BA-∗SA-1.27R

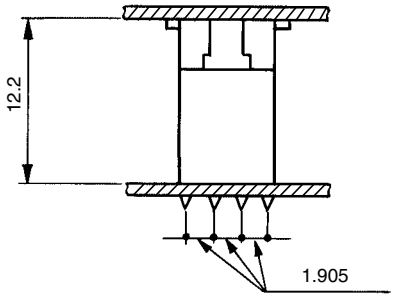
◆SMT Type Application Pattern

FX2-∗P-1.27SV(L)



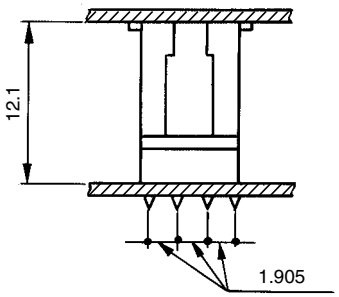
FX2-∗S-1.27SV(L)

FX2-∗S-1.27SV(L)



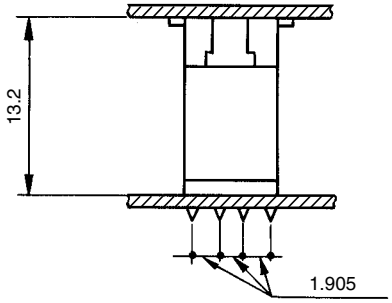
FX2C(A)-∗P-1.27DSA(L)

FX2-∗P-1.27SV(L)



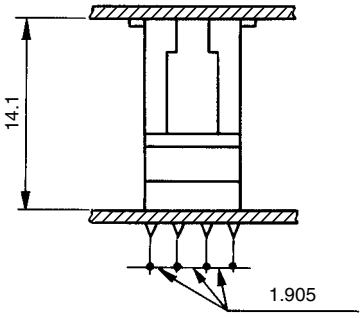
FX2C(A)-∗S-1.27-DSA(L)

FX2-∗S-1.27SV(L)



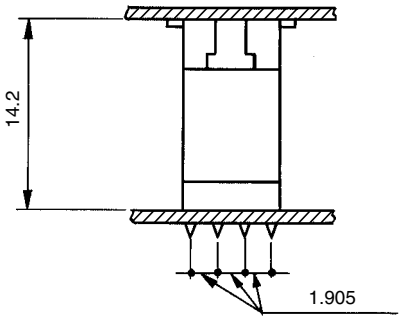
FX2C(A)1-∗P-1.27DSA(L)

FX2-∗P-1.27SV(L)



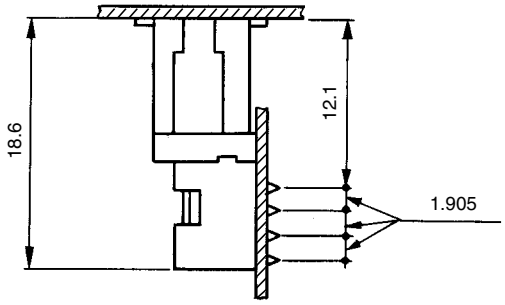
FX2C(A)2-∗S-1.27DSA(L)

FX2-∗S-1.27SV(L)



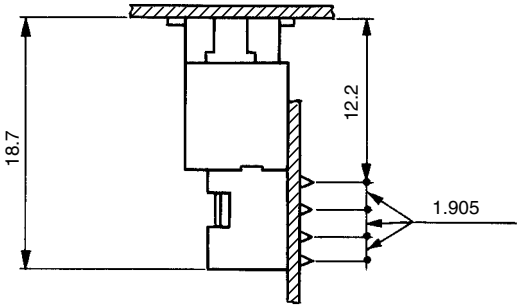
FX2C(A)2-∗P-1.27DSA(L)

FX2-∗P-1.27SV(L)



FX2-∗S-1.27DS(L)

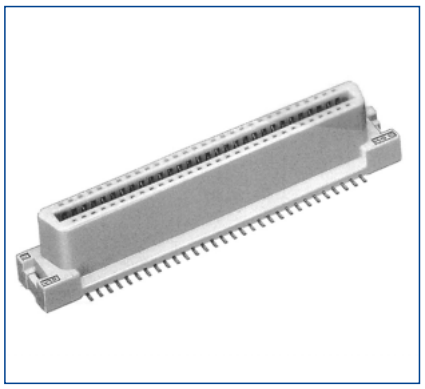
FX2-∗S-1.27SV(L)

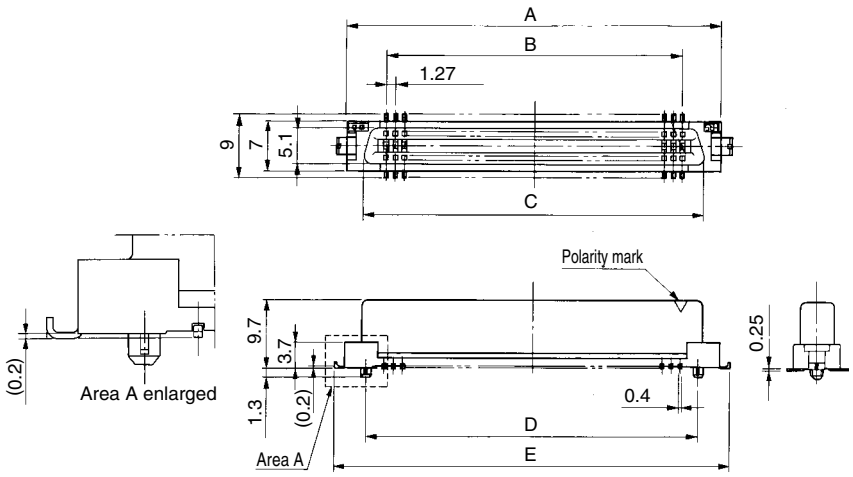


FX2-∗P-1.27DS(L)

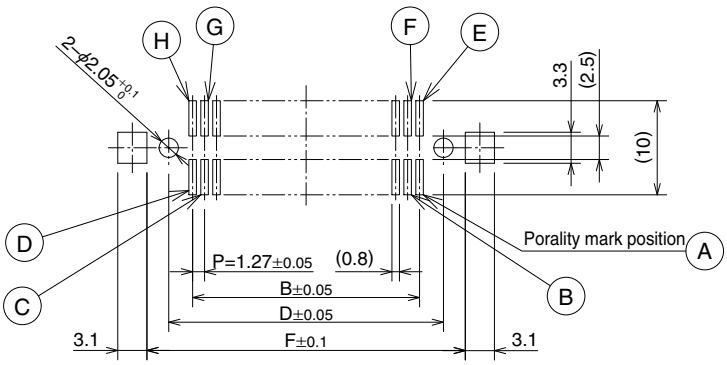
■Receptacle SMT Type

●Straight Type





◆PCB mounting pattern



n=Number of Contacts

(A)	a1	(E)	b1
(B)	a2	(F)	b2
(C)	$a\frac{n}{2}-1$	(G)	$b\frac{n}{2}-1$
(D)	$a\frac{n}{2}$	(H)	$b\frac{n}{2}$

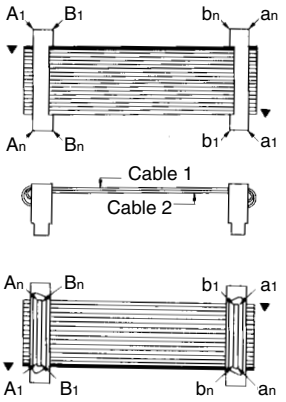
Unit : mm

Part No.	HRS No.	No. of Contacts	A	B	C	D	E	F	RoHS
FX2-20S-1.27SV(71)	572-2101-3 71	20	22.75	11.43	17.63	16.51	——	——	Yes
FX2-20S-1.27SVL(71)	572-2151-1 71						25.75	21.15	
FX2-32S-1.27SV(71)	572-2102-6 71	32	30.37	19.05	25.25	24.13	——	——	
FX2-32S-1.27SVL(71)	572-2152-4 71						33.37	28.77	
FX2-40S-1.27SV(71)	572-2103-9 71	40	35.45	24.13	30.33	29.21	——	——	
FX2-40S-1.27SVL(71)	572-2153-7 71						38.45	33.85	
FX2-52S-1.27SV(71)	572-2104-1 71	52	43.07	31.75	37.95	36.83	——	——	
FX2-52S-1.27SVL(71)	572-2154-0 71						46.07	41.47	
FX2-60S-1.27SV(71)	572-2105-4 71	60	48.15	36.83	43.03	41.91	——	——	
FX2-60S-1.27SVL(71)	572-2155-2 71						51.15	46.55	
FX2-68S-1.27SV(71)	572-2106-7 71	68	53.23	41.91	48.11	46.99	——	——	
FX2-68S-1.27SVL(71)	572-2156-5 71						56.23	51.63	
FX2-80S-1.27SV(71)	572-2107-0 71	80	60.85	49.53	55.73	54.61	——	——	
FX2-80S-1.27SVL(71)	572-2157-8 71						63.85	59.25	
FX2-100S-1.27SV(71)	572-2108-2 71	100	73.55	62.23	68.43	67.31	——	——	
FX2-100S-1.27SVL(71)	572-2158-0 71						76.55	71.95	
FX2-120S-1.27SV(71)	572-2109-5 71	120	86.25	74.93	81.13	80.01	——	——	
FX2-120S-1.27SVL(71)	572-2159-3 71						89.25	84.65	

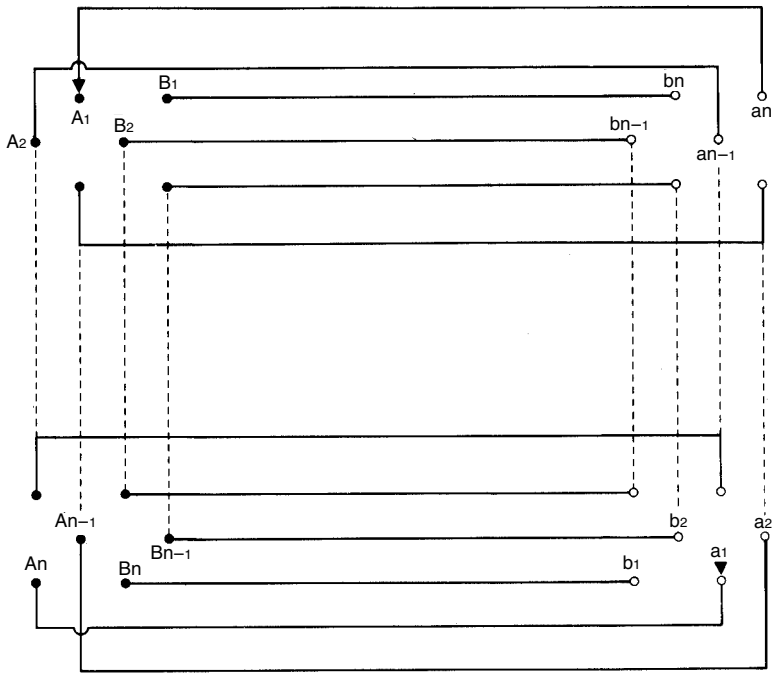
Recommended Circuit Design

$$n = \frac{\text{Number of Contacts}}{2}$$

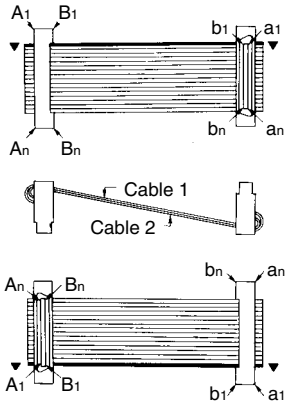
Type A, B



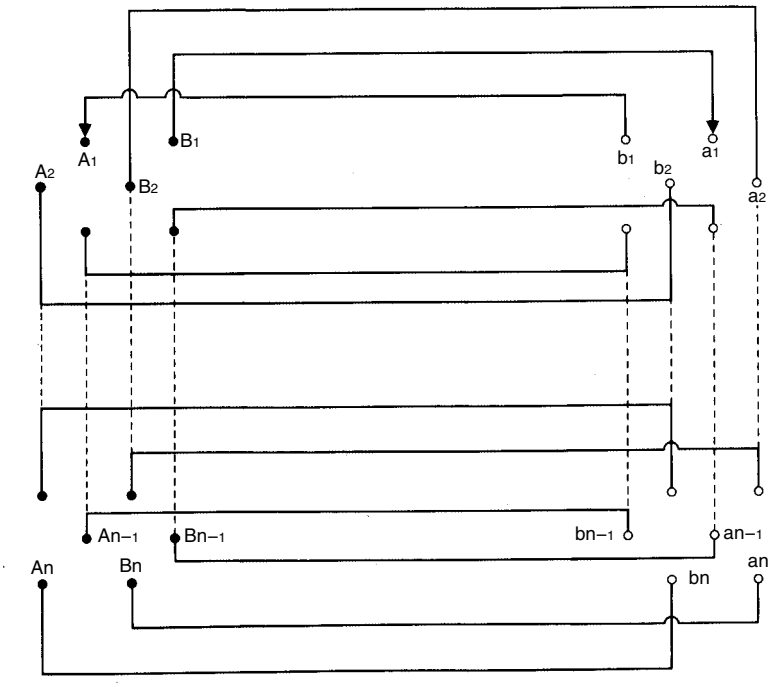
Cable 1 connection	Cable 2 connection
A1 — an	B1 — bn
A2 — an-1	B2 — bn-1
⋮ ⋮	⋮ ⋮
An — a1	Bn — b1



Type A, A



Cable 1 connection	Cable 2 connection
A1 — b1	B1 — a1
A2 — b2	B2 — a2
⋮ ⋮	⋮ ⋮
An — bn	Bn — an



◆How to use the connector

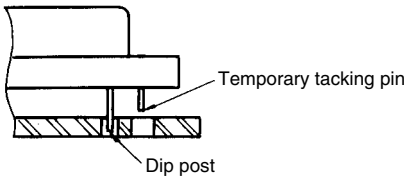
1. Selection of connectors

①FX2 Series connectors are designed to prevent flux creep-up in the soldering operation of straight type products, and the user needs to select from either non-cleaning types (FX2C-∗ and FX2B-∗) or cleaning types(FX2CA-∗ and FX2BA-∗). Please make sure to clean the right-angle types(FX2-∗ and FX2B-∗). When cleaning them, be sure to use clean liquids.

②When using a socket cable type, the user needs to select a locked type if a load is applied on vibration/impact, or the cable. Additionally, please be sure to use cabler clamps when a load is applied on the cable.

2. The process of temporary tacking on the PCB

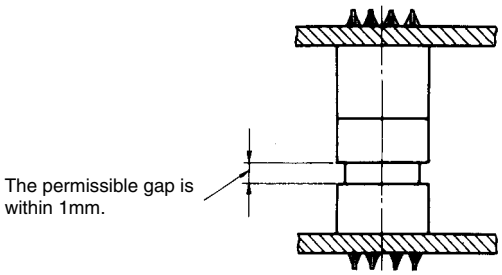
Please use the types equipped with temporary tacking pins when a temporary tacking process is required for mounting on the PCB.(Qualified thickness of the PCB : t=1.6±0.1)



(Precaution) : In order to prevent damage to the dip post when it is mounted on the PCB, conduct the inserting operation while keeping the PCB and the connectors parallel to one another while the temporary tacking pin is pushed in after the dip post has been guided into the through hole of the PCB.

3. Mating Side Tolerance Clearance

(Example) In the case of FX2C-ĐP-1.27DSA and FX2C-ĐS-1.27DSA∗P-1.27DSA and FX2C-∗S-1.27DSA



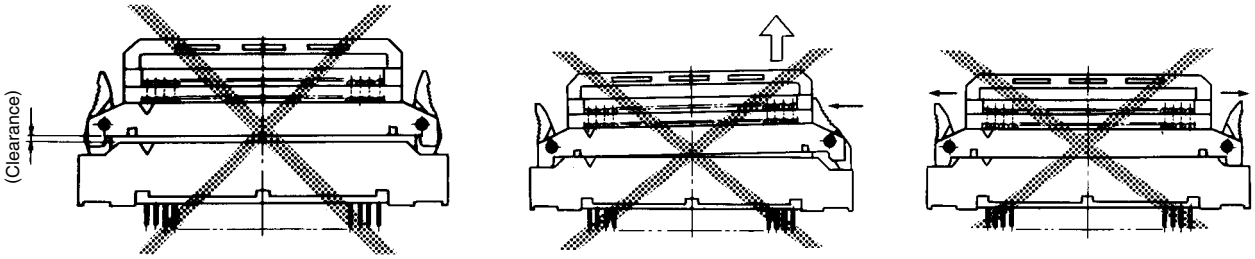
The permissible gap when a male and a female connectors are mated together shall be within 1 mm. (excluding locked cable types)

4. Indtructions for using locked cable types

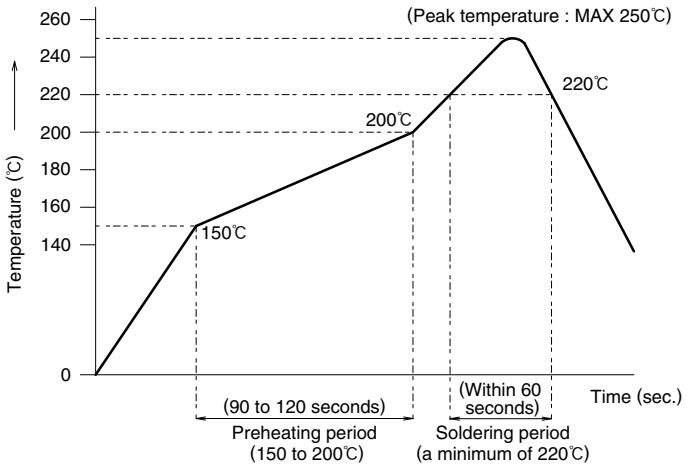
①When mating connectors, please insert them to the end so that there is no gap in-between.

②When pulling out a connector, please be sure to pull it out after turning in the lock levers on both sides.
If you pull out the connector while turning in only the lever on one side, the connector could be damaged.

③Please make sure that the lock lever will not be open to the outside.



5. Temperature profile for mounting SMT type connectors (reference)



<Applicable conditions>

Reflow method : Reflow together with far infrared ray and hot air (SENSBEY NR-II)

Environment in the reflow furnace : Ambient air

Solder : Cream type (Sn-3Ag-0.5Cu)
(Flux contained amount: 11 Wt%)
(M705-221CM5-42-10.5 manufactured by Senju Metal Industry Co., Ltd.)

PCB : Size 110 x 40 x 1.6 mm
Material: Glass epoxy

Metal mask thickness : 0.2 mm (straight type)
0.15 mm (right-angled type)

Note 1 : This temperature profile shows recommended values. Temperature should be measured on the top of the connector.
Note 2 : The number of reflow processes should be twice or less.
Note 3 : Impacts due to the type and manufacture of solder cream, PCB size, and other mounting materials, etc, may differ depending on the conditions. Please be sure to confirm the mounting conditions before use.

∗ When using a 0.15 mm metal mask for the straight type connectors, please contact our technology center for recommended apertures.

6. Dimensions between the PCBs for SMT type connectors

The gap between the PCBs for male and female connectors shown in the applications does not include the solder cream thickness. Therefore, please take into account the thickness of the solder cream so that it is added to the gap between the PCBs after mounting the connectors.

USA:
HIROSE ELECTRIC (U.S.A.), INC. HEADQUARTERS
2688 Westhills Court, Simi Valley, CA 93065-6235
Phone : +1-805-522-7958
Fax : +1-805-522-3217
http://www.hirose.com/us/

USA:
HIROSE ELECTRIC (U.S.A.), INC. SAN JOSE OFFICE
3255 Scott Boulevard, Building 7, Suite 101
Santa Clara, CA 95054
Phone : +1-408-253-9640
Fax : +1-408-253-9641
http://www.hirose.com/us/

USA:
HIROSE ELECTRIC (U.S.A.), INC. CHICAGO OFFICE
580 Waters Edge Lane, Suite 205 Lombard IL 60148
Phone : +1-630-282-6701
Mail : inquiries@hirose.com
http://www.hirose.com/us/

USA:
HIROSE ELECTRIC (U.S.A.), INC. DETROIT OFFICE (AUTOMOTIVE)
17197 N. Laurel Park Drive, Suite 253, Livonia, MI 48152
Phone : +1-734-542-9963
Fax : +1-734-542-9964
http://www.hirose.com/us/

THE NETHERLANDS:
HIROSE ELECTRIC EUROPE B.V.
Hogehillweg #8 1101 CC Amsterdam Z-O
Phone : +31-20-6557460
Fax : +31-20-6557469
http://www.hirose.com/eu/

GERMANY:
HIROSE ELECTRIC EUROPE B.V. GERMAN BRANCH
Herzog-Carl-Strasse 4 D-73760 Ostfildern (Scharnhauser Park)
Phone : +49-711-4560-02-1
Fax : +49-711-4560-02-299
http://www.hirose.com/eu/

GERMANY:
HIROSE ELECTRIC EUROPE B.V. NUERNBERG OFFICE
Muggenhofer Str. 136 90429 Nuernberg
Phone : +49-911 32 68 89 63
Fax : +49-911 32 68 89 69
http://www.hirose.com/eu/

GERMANY:
HIROSE ELECTRIC EUROPE B.V. HANOVER OFFICE
Bayernstr. 3, Haus C 30855 Langenhagen, Germany
Phone : +49-511 97 82 61 30
Fax : +49-511 97 82 61 35
http://www.hirose.com/eu/

FRANCE:
HIROSE ELECTRIC EUROPE B.V. PARIS OFFICE
Regus La Garenne Colombes,Place de La Belgique, 71 Boulevard National La Garenne Colombes, 92250, France
Phone : +33 (0) 1 7082 3170
Fax : +33 (1) 7082 3101
http://www.hirose.com/eu/

UNITED KINGDOM:
HIROSE ELECTRIC EUROPE BV (UK BRANCH)
4 Newton Court, Kelvin Drive, Knowlhill, Milton Keynes, MK5 8NH
Phone : +44-1908 202050
Fax : +44-1908 202058
http://www.hirose.com/eu/

CHINA:
HIROSE ELECTRIC (SHANGHAI) CO., LTD.
1601, Henderson Metropolitan, NO.300, East Nanjing Road, Huangpu District, Shanghai, China 200001
Phone : +86-21-6391-3355
Fax : +86-21-6391-3335
http://www.hirose.com/cn/

CHINA:
HIROSE ELECTRIC (SHANGHAI) CO.,LTD. BEIJING BRANCH
A1001, Ocean International Center, Building 56# East 4th Ring Middle Road, ChaoYang District, Beijing, 100025
Phone : +86-10-5165-9332
Fax : +86-10-5908-1381
http://www.hirose.com/cn/

CHINA:
HIROSE ELECTRIC TECHNOLOGIES (SHENZHEN) CO., LTD.
Room 09-13, 19/F, Office Tower Shun Hing Square, Di Wang Commercial Centre, 5002 Shen Nan Dong Road, Shenzhen City, Guangdong Province, 518008
Phone : +86-755-8207-0851
Fax : +86-755-8207-0873
http://www.hirose.com/cn/

HONG KONG:
HIROSE ELECTRIC HONGKONG TRADING CO., LTD.
Room 1001, West Wing, Tsim Sha Tsui Centre, 66 Mody Road, Tsim Sha Tsui East, Kowloon, Hong Kong
Phone : +852-2803-5338
Fax : +852-2591-6560
http://www.hirose.com/hk/

TAIWAN:
HIROSE ELECTRIC TAIWAN CO., LTD.
103 8F, No.87, Zhengzhou Rd., Taipei
Phone : +886-2-2555-7377
Fax : +886-2-2555-7350
http://www.hirose.com/tw/

KOREA:
HIROSE KOREA CO., LTD.
250, Huimanggongwon-ro, Siheung-si, Gyeonggi-do, Korea, 429-849
Phone : +82-31-496-7000,7124
Fax : +82-31-496-7100
http://www.hirose.co.kr/

SINGAPORE:
HIROSE ELECTRIC SINGAPORE PTE. LTD.
10 Anson Road #26-16, International Plaza 079903, Singapore
Phone : +65-6324-6113
Fax : +65-6324-6123
http://www.hirose.com/sg/

INDIA:
HIROSE ELECTRIC SINGAPORE PTE. LTD. DELHI LIAISON OFFICE
Office NO.519, Regus-Green Boulevard, Level5, Tower C, Sec62, Plot B-9A, Block B, Noida, 201301, Uttar Pradesh, India
Phone : +91-12-660-8018
Fax : +91-120-4804949
http://www.hirose.com/sg/

INDIA:
HIROSE ELECTRIC SINGAPORE PTE. LTD. BANGALORE LIAISON OFFICE
Unit No-403, 4th Floor, No-84, Barton Centre, Mahatma Gandhi (MG) Road, Bangalore 560 001, Karnataka, India
Phone : +91-80-4120 1907
Fax : +91-80-4120 9908
http://www.hirose.com/sg/

MALAYSIA:
HIROSE ELECTRIC SINGAPORE PTE. LTD.
1-10-07, Suntech @ Penang Cybercity (1164),Lintang Mayang Pasir 3,11950, Bayan Baru, Penang, Malaysia.
Phone : +604-619-2564
Fax : +604-619-2574
http://www.hirose.com/sg/

THAILAND:
HIROSE ELECTRIC SINGAPORE PTE. LTD.
BANGKOK OFFICE (REPRESENTATIVE OFFICE)
Unit 4703, 47th FL., 1 Empire Tower, South Sathorn Road, Yannawa, Sathorn, Bangkok 10120 Thailand
Phone : +66-2-686-1255
Fax : +66-2-686-3433
http://www.hirose.com/sg/

Creative Links to World Electronics ®

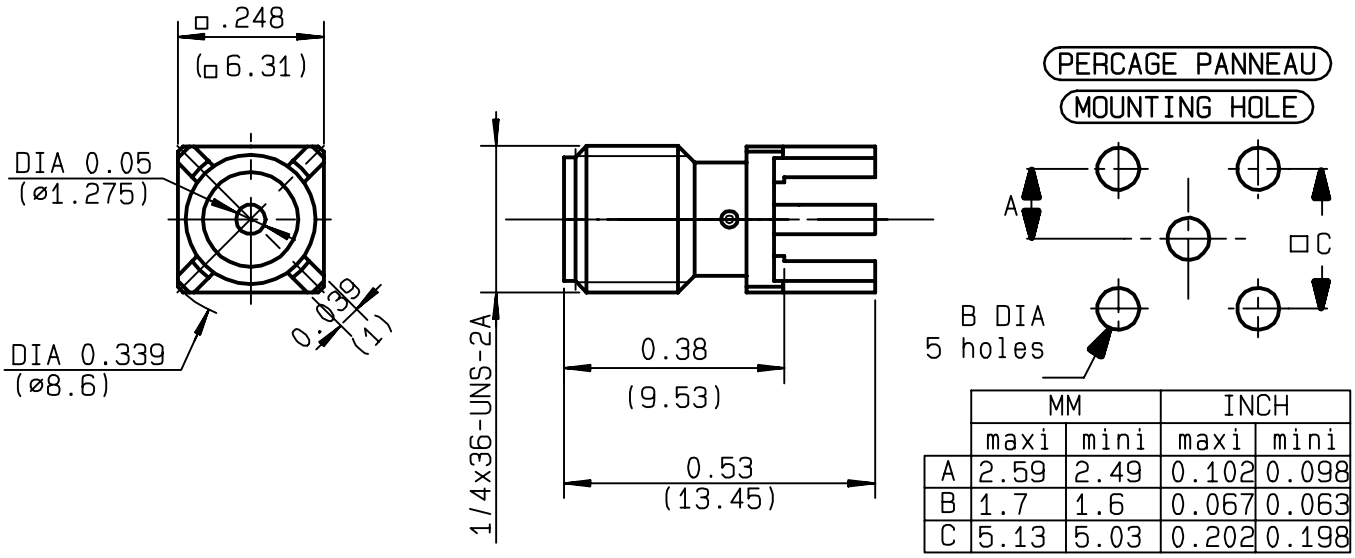
HRS

HIROSE ELECTRIC CO.,LTD.
2-6-3,Nakagawa Chuoh,Tsuzuki-Ku,Yokohama-Shi 224-8540,JAPAN
TEL: +81-45-620-3526 Fax: +81-45-591-3726
http://www.hirose.com
http://www.hirose-connectors.com

STRAIGHT JACK RECEPTACLE FOR PCB SOLDER LEGS

R125.426.000

SERIES SMA



	MM		INCH	
	maxi	mini	maxi	mini
A	2.59	2.49	0.102	0.098
B	1.7	1.6	0.067	0.063
C	5.13	5.03	0.202	0.198

NOMINAL IMPEDANCE	**50**	Ω
FREQUENCY RANGE	**0-18**	GHz
TEMPERATURE RATING	**-65/+165**	°C
V.S.W.R	**1.04** + **.0045** x F(GHz)Maxi	
RF INSERTION LOSS	**0.05** $\sqrt{F}$(GHz) dB Maxi	
VOLTAGE RATING	**500**	Veff Maxi
DIELECTRIC WITHSTANDING VOLTAGE	**1000**	Veff Mini
INSULATION RESISTANCE	**5000**	MΩMini
HERMETIC SEAL	**NA**	$Atm.cm^3/s$
LEAKAGE (pressurized only)	**NA**	
MECHANICAL DURABILITY	**500**	Cycles
WEIGHT	**1.24**	gr
SPECIFICATION		

CABLES :

OTHERS CHARACTERISTICS

CABLE RETENTION	**NA**	N Mini
CENTER CONTACT RETENTION		
Axial force - mating end	**27**	N Mini
Axial force - opposite end	**27**	N Mini
Torque	**2.8**	cm.N Mini
RECOMMENDED TORQUES		
Mating	**NA**	cm.N
Panel nut	**NA**	cm.N
Clamp nut	**NA**	cm.N

CONNECTOR PARTS	MATERIALS	FINISH (all values are given in micrometers)
BODY	STAINLESS STEEL	GOLD 0.5 OVER NICKEL 2
OUTER CONTACT		
CENTER CONTACT	BERYLLIUM COPPER	GOLD 1.3 OVER NICKEL 2
INSULATOR	PTFE	-
GASKET		-
OTHERS PIECES		

ISSUE	CREATION DATE	FILE PART-NUMBER
9945D01	14/06/1995	

RADIALL®

TRIQUES E.

Connect to the future