UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Universitat Politècnica de València

Escuela Técnica Superior de Ingenieros de Telecomunicación

Departamento de Ingeniería Electrónica

Estudio e implementación de un sistema de caracterización automática de circuitos integrados analógicos

Ingeniería de Telecomunicación

Autor: Rafael Pérez Carmona

Director Marvell: Antonio Pairet Molina

Director UPV: Vicente Herrero Bosch

València, 2014

Agradecimientos

Después de estos bonitos años de carrera, no podría culminarla sin dedicar estas líneas a las personas que me han acompañado y apoyado a lo largo de ella.

Deseo hacer una mención especial a mis padres y a mi hermano por ser las personas que directamente me han aguantado en los buenos y en los malos momentos. Sin ellos, esto no hubiese sido posible. ¡SIEMPRE OS LO AGRADECERÉ!

También deseo mencionar a mis abuelos y en especial mi abuela Amparo que les hubiera gustado verme con este sueño cumplido. A mis tíos y primos en especial a Carol que sé que se alegran mucho de esto.

No menos importante es Raquel, que se ha incorporado en este último tramo y que es la pieza que faltaba en este puzzle.

A mis amigos, los de toda la vida y los formados en la universidad.

Por último a Marvell Hispania por darme la oportunidad de realizar este proyecto y en especial a Antonio Pairet y Vicente Herrero por intentar hacer de mí un buen profesional.

A todos vosotros, ¡MUCHÍSIMAS GRACIAS!

Resumen

El objetivo de este proyecto es implementar un sistema capaz de ejecutar automáticamente un DVT en un circuito integrado analógico. Un DVT es un programa intensivo de pruebas que se realiza para verificar que se cumplen todas las especificaciones del producto.

Este sistema es capaz de realizar un DVT en base a los casos de test definidos por el usuario y al finalizar realizar un informe final organizado con todos los resultados obtenidos.

El rendimiento del sistema se pone de manifiesto cuando hay que verificar diversos chips con varios casos de test como puede ser con distintas temperaturas, distintas alimentaciones, etc. El elevado coste temporal de realizar esta verificación nos ha llevado a la automatización de este programa de pruebas. De esta manera, podemos configurar los casos de test que se quieren evaluar y así poder centrar nuestros esfuerzos en otras labores.

El proyecto ha sido desarrollado en Marvell Hispania S.L. como parte del programa de pruebas del front end analógico del producto G.hn de Marvell para comunicaciones de banda ancha sobre powerline.

Lista de acrónimos

AFE	Analog Front End
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
DUT	Device Under Test
DVT	Design Verification Test
GPIB	General Purpose Instrumentation Bus
GTK	GNOME ToolKit
GUI	Graphical User Interface
HPIB	Hewlett-Packard Interface Bus
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
LXI	LAN eXtensions for Instrumentation
MSB	Most Significant Bit
PLC	PowerLine Communications
PXI	PCI eXtensions for Instrumentation
RAD	Rapid Application Development
SCPI	Standard Commands for Programmable Instruments
SPI	Serial Peripheral Interface
SVB	Silicon Validation Board
USB	Universal Serial Bus
VCP	Virtual COM Port
VXI	VME eXtensions for Instrumentation

Cuadro 1: Tabla de acrónimos

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

La fabricación de circuitos integrados es un proceso muy delicado. Es lógico pensar que una vez tenemos el circuito integrado fabricado hay que verificar que se comporta de acuerdo a la simulación y que las pequeñas variaciones que pueda sufrir el proceso de fabricación no afectan al funcionamiento del mismo.

Esta necesidad de verificar físicamente el circuito integrado nos lleva a tener que realizar muchas comprobaciones in situ con estímulos reales para verificar que la respuesta a estos estímulos es la esperada.

El DVT (*Design Verification Test*) es el encargado de ejecutar una serie de tests para caracterizar el circuito integrado y verificar que tras el proceso de fabricación el circuito integrado se comporta tal y como era de esperar.

Con el objetivo de cubrir esta necesidad, se ha realizado este proyecto final de carrera que pretende ser una herramienta software para la ejecución automática del DVT.

1.2. Marco del proyecto final de carrera

Este proyecto final de carrera ha sido llevado a cabo en un entorno empresarial, en concreto en la empresa Marvell Hispania situada en el Parque Tecnológico de Valencia y consta del desarrollo de una herramienta software que permite verificar las especificaciones del front end analógico del producto G.hn de Marvell para comunicaciones de banda ancha sobre powerline.

La actividad de Marvell Hispania es la comercialización del AFE, el circuito integrado digital, el diseño de referencia y el software para el producto G.hn.

Marvell ofrece estos productos a otras compañías que se encargan de realizar la fabricación final del módem PLC (*Power Line Communications*) y la comercialización del producto bajo su nombre. Estas compañías comercializan el producto bajo su nombre pero se ven obligadas a comprar los circuitos integrados a las empresas que los diseñan.

La solución de Marvell ofrece velocidades de transmisión muy elevadas permitiendo la distribución de datos, audio y vídeo dentro del entorno doméstico sin provocar cuellos de botella en la transmisión. Ello supone una revolución en el mundo de las comunicaciones al poder usar el enchufe eléctrico para conectarse a la red local y poder disfrutar de todas las ventajas que esto ofrece, todo ello a muy bajo coste al utilizar la red eléctrica ya existente.

Los circuitos integrados diseñados por Marvell se engloban dentro del estándar G.hn regulado por el HomeGrid Forum y el objetivo es proporcionar velocidades de hasta 1 Gbps en las líneas de bajo y medio voltaje compitiendo con las actuales redes domésticas convencionales (Cable, Wi-Fi, etc.).

1.3. Objetivos

Dentro del marco empresarial que nos encontramos y con la finalidad de cubrir las necesidades comentadas anteriormente, se ha realizado el presente proyecto final de carrera con el objetivo de poder caracterizar y verificar unas muestras de chips analógicos que la fundición nos ha proporcionado para validar las especificaciones del diseño en diferentes casos. Una vez obtenidos los resultados y tras su validación se da comienzo a la fabricación en masa.

El objetivo concreto es la implementación de un DVT capaz de obtener medidas del chip en función de la temperatura, la alimentación y la propia configuración. Estas medidas serán almacenadas en ficheros ASCII e imágenes individuales para cada test. Al finalizar, el DVT genera un informe final con toda la información organizada. Los tests implementados son los siguientes:

- Puntos de reposo
- Densidad Espectral de Ruido
- Función de Transferencia
- Consumo dinámico
- MTPR (MultiTone Power Ratio)

A todo esto hay que sumarle la implementación de una GUI (*Graphical User Interface*) que nos permite realizar medidas independientes de forma sencilla para casos puntuales de test. Todo esto sin tener que crear un fichero de configuración para el DVT.

1.4. Estado del arte

Según [6], la verificación se encuentra alrededor de un 60 %-70 % del coste del proyecto en términos humanos, de computación y de tiempo. Esto significa que más de la mitad de los recursos humanos, del tiempo empleado y de los recursos informáticos son usados para la verificación.

En el marco empresarial que nos encontramos, disponemos de algunos de los tests anteriores automatizados pero sin interconexión entre ellos. Esto quiere decir que la interacción humana debe realizarse frecuentemente para realizar el DVT.

La poca interconexión entre las aplicaciones y la necesidad de obtener licencias del software nos ha motivado a implementar el set de tests con un lenguaje de programación abierto como es Python. La decisión de emplear este lenguaje ha sido la posibilidad de integrar esta herramienta dentro de la plataforma de validación de Marvell Hispania.

La plataforma de validación empleada (Robot Framework) permite la automatización de pruebas mediante la programación de keywords. Las keywords son una serie de ficheros tabulados donde se realizan llamadas a otras keywords o a funciones de bajo nivel escritas en Python o Java.

De otra manera, Robot Framework es un intérprete de código fuente tabulado y los ficheros fuente hacen llamadas a keywords incluidas en el intérprete, keywords creadas por el usuario y funciones de bajo nivel creadas por el usuario.

En la figura 1.1 se puede ver el IDE (*Integrated Development Environment*) de Robot Framework.

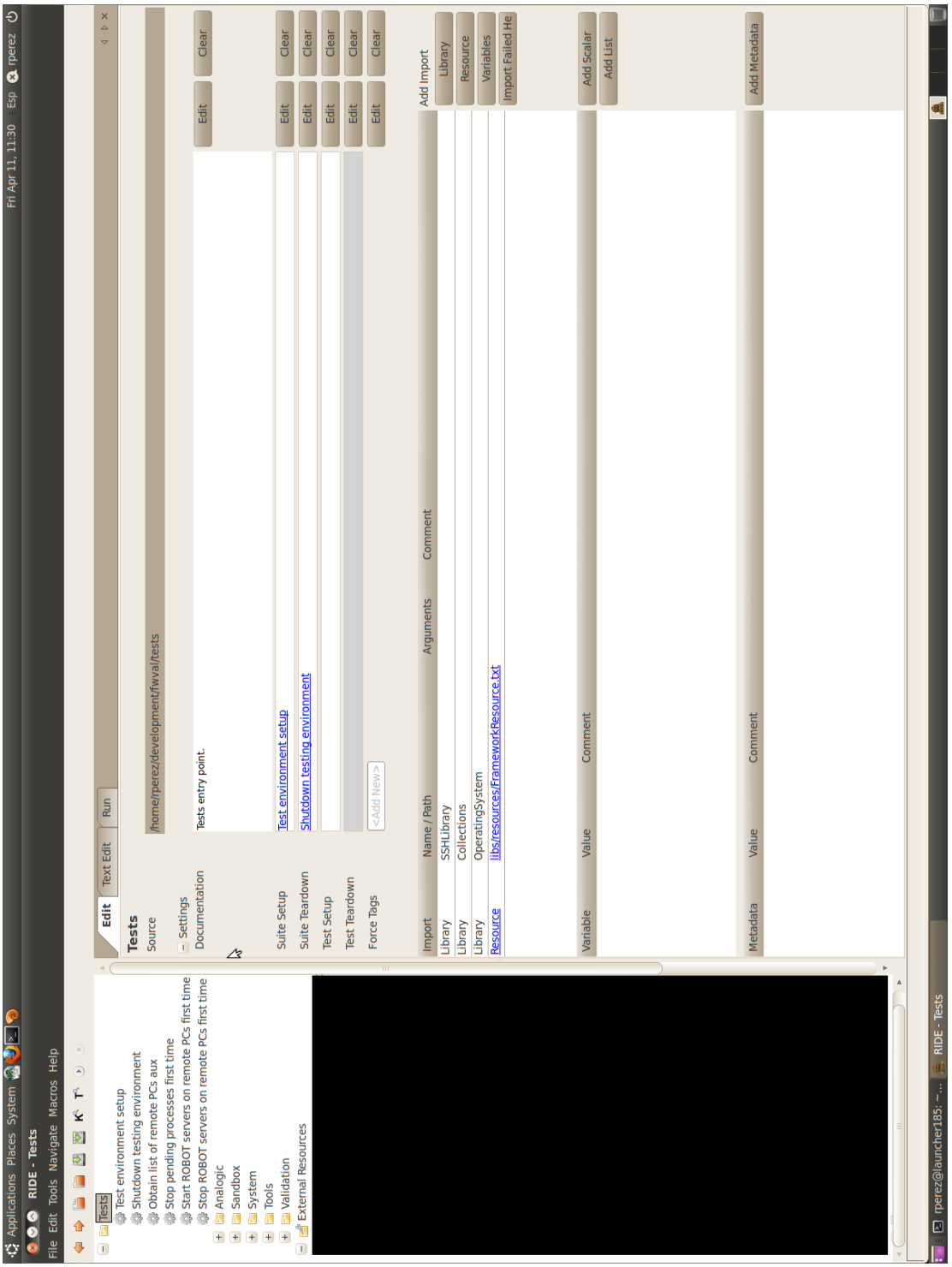


Figura 1.1: IDE de Robot Framework

1.5. Estructura de capítulos

El presente proyecto está estructurado en los siguientes capítulos:

- Capítulo 2. Descripción funcional del sistema.
- Capítulo 3. Descripción funcional y algorítmica del test Puntos de reposo.
- Capítulo 4. Descripción funcional y algorítmica del test Densidad Espectral de Ruido.
- Capítulo 5. Descripción funcional y algorítmica del test Función de Transferencia.
- Capítulo 6. Descripción funcional y algorítmica del test Consumo dinámico.
- Capítulo 7. Descripción funcional y algorítmica del test MTPR.
- Capítulo 8. Descripción funcional y algorítmica del módulo SPI master.
- Capítulo 9. Descripción funcional y algorítmica de la GUI.
- Capítulo 10. Conclusiones y líneas futuras.
- Apéndice A. Códigos fuente más importantes.
- Apéndice B. Descripción del fichero settings_dvt.py.
- Apéndice C. Librerías de instrumentación.
- Apéndice D. Código fuente del SPI master.
- Apéndice E. Código fuente de la GUI.
- Apéndice F. Ejemplo de informe final generado por el DVT.

Capítulo 2

Descripción general del sistema

2.1. Introducción y descripción general del DUT

A efectos prácticos, el DUT (*Device Under Test*) está compuesto por dos canales de transmisión (TXA, TXB) y dos canales de recepción (RXA, RXB).

Como hemos comentado anteriormente, el DUT está regulado por el estándar G.hn que establece, entre otras cosas, que la modulación empleada en comunicaciones powerline sea OFDM (*Orthogonal Frequency-Division Multiplexing*) donde cada subportadora transmite una constelación QAM de 12 bits (4096-QAM). Además establece que el código tenga un control de errores FEC (*Forward Error Correction*) y LDPC (*Low-Density Parity-Check*).

2.2. Descripción general del DVT

El objetivo del DVT es obtener una caracterización lo más completa posible del circuito integrado analógico con la finalidad de verificarlo y asegurarnos que se cumplen las especificaciones.

El algoritmo que representa la ejecución de un DVT completo es el siguiente:

```
for TEMPERATURA do
    for ALIMENTACIÓN do
        for GANANCIA (Configuración del circuito integrado) do
            Ejecuta Puntos de reposo
            Ejecuta Densidad Espectral de Ruido
            Ejecuta Función de Transferencia
            Ejecuta Consumo dinámico
            Ejecuta MultiTone Power Ratio (MTPR)
        end for GANANCIA
    end for ALIMENTACIÓN
end for TEMPERATURA
```

En el fichero de configuración del DVT el usuario es el encargado de definir mediante una serie de vectores qué rango de temperaturas, alimentaciones y ganancias son objeto de caracterización.

2.3. Arquitectura de test

Debido a que nos encontramos en un marco empresarial, ha sido prácticamente obligado la inserción del DVT dentro de la arquitectura de test de la empresa. Esta inserción ha sido realizada por diversos motivos:

- Utilización del mismo framework que en validación firmware (integración)
- Reutilización de código
- Único mantenimiento de la arquitectura
- En caso de actualización de librerías de bajo nivel no tenemos que cambiar nuestro código fuente

La arquitectura que da soporte al DVT se muestra en la figura 2.1

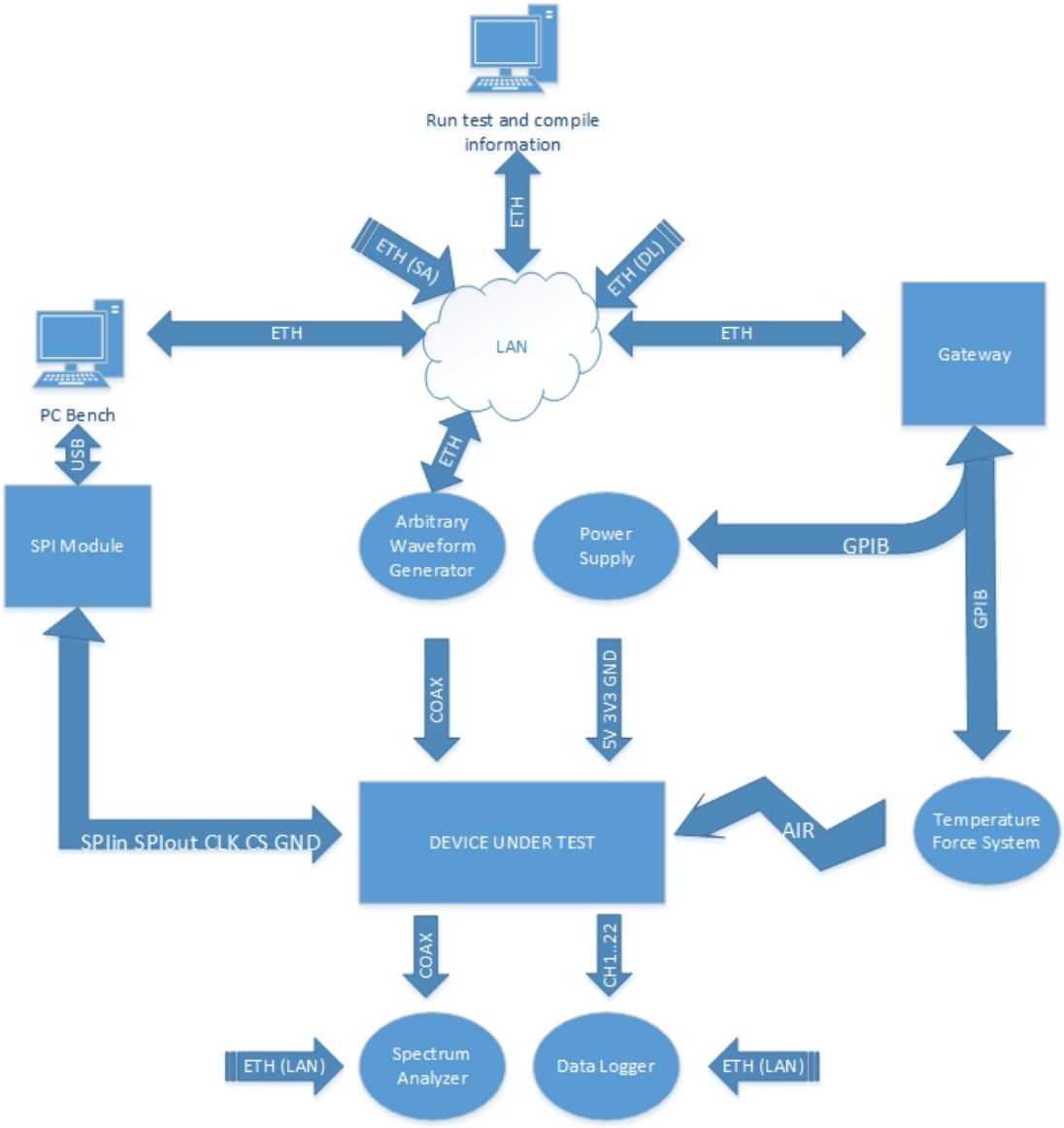


Figura 2.1: Arquitectura de test

2.4. Estructura de ficheros

Con la finalidad de hacer un código fácilmente reutilizable, se han creado librerías independientes para el control remoto de instrumentos. De este modo, cualquier usuario puede reutilizar estas librerías sin afectar al código del DVT. También ha sido necesario la creación de ficheros auxiliares para particularizar ciertos ajustes al DVT. En la figura 2.2 se puede apreciar la estructura de ficheros que forma el DVT.

2.5. Conexiones del DUT

El DUT es un AFE compuesto por dos canales de transmisión y dos canales de recepción. El objetivo es la caracterización de estos canales mediante la ejecución de los tests citados anteriormente que en capítulos posteriores serán descritos.

Debido a que el DUT se trata de un circuito integrado, tenemos que recurrir al uso de una SVB (*Silicon Validation Board*) para poder realizar las conexiones de una forma cómoda así como fijar los valores de carga de los canales.

En la figura 2.3 se pueden observar detalladamente las conexiones necesarias para el DVT.

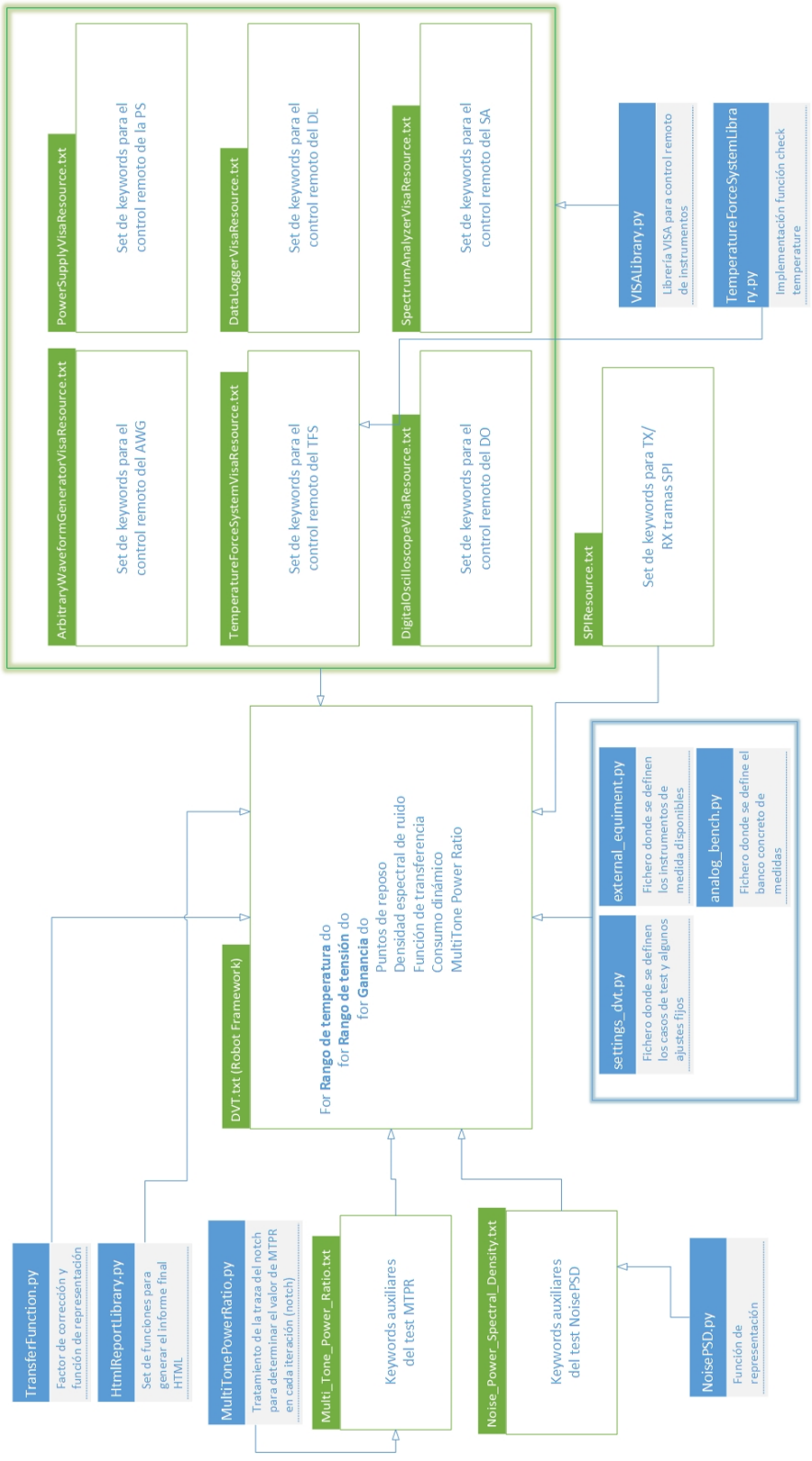


Figura 2.2: Estructura de ficheros

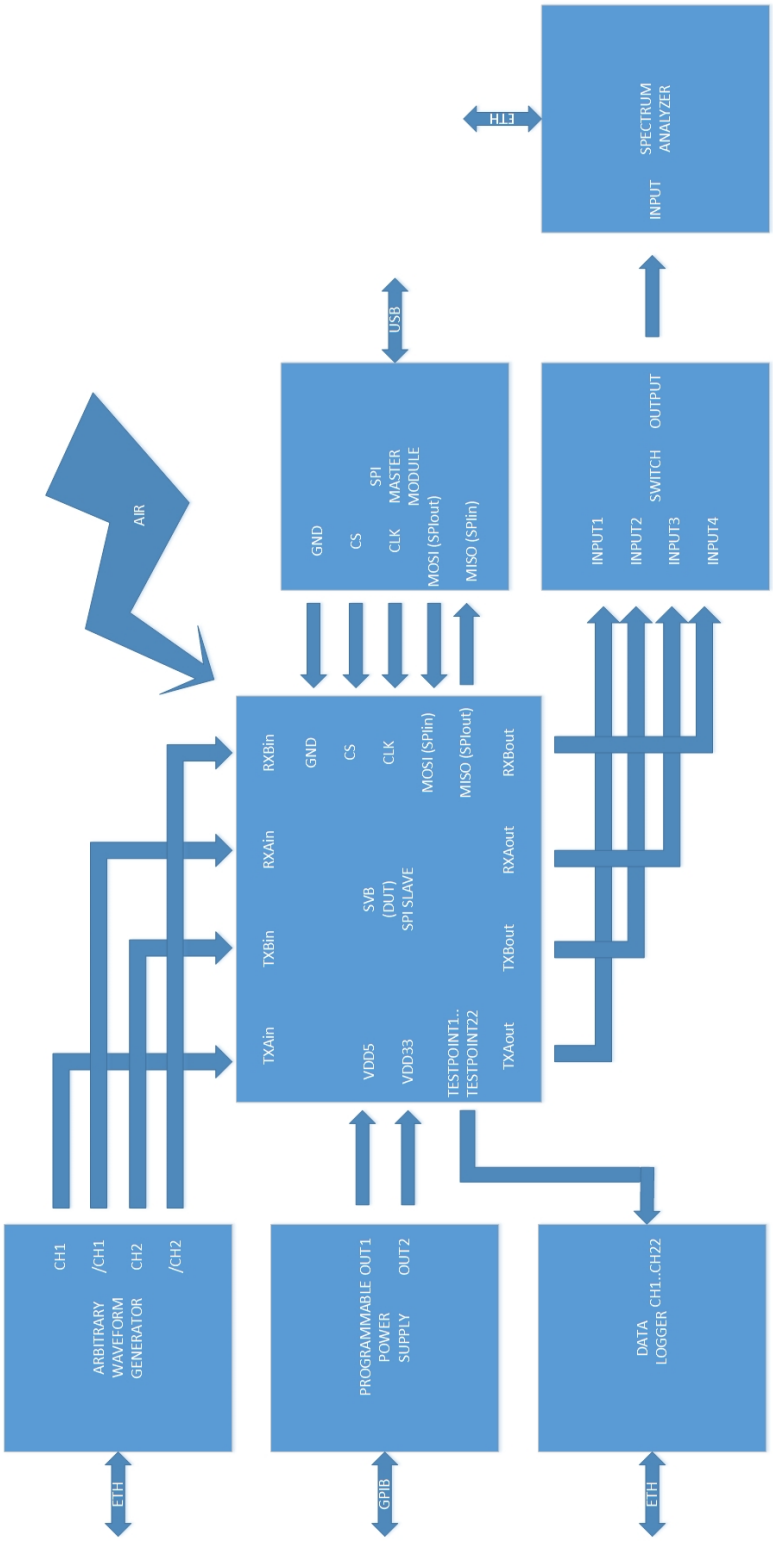


Figura 2.3: Conexiones SVB

2.6. Control Remoto

Es lógico pensar que para poder alcanzar el objetivo que nos hemos propuesto es necesario que la instrumentación pueda ser programada. El control remoto de la instrumentación es una de las necesidades primordiales para abordar el presente objetivo.

Los instrumentos que nos atañen se pueden clasificar en dos grupos (en función del bus de conexión):

- Dispositivos LXI (red ethernet)
  - Generador de señales arbitrarias
  - Analizador de espectro
  - Data logger (multímetro multicanal)
- Dispositivos GPIB
  - Fuente de alimentación
  - Sistema de forzado de temperatura

Existen otros tipos de buses para control remoto como es VXI, PXI, etc. Debido a diversas causas como diferencia de edad entre los instrumentos, diferentes fabricantes, etcétera existe esta variedad.

La tendencia natural es que la instrumentación se convierta en equipos de red ethernet. Esto es un avance muy importante por el hecho de que buses como GPIB, VXI y otros tengan muchas limitaciones en cuanto a número máximo de instrumentos, longitud máxima del bus, dificultad para control remoto fuera del propio bus, etc.

Para poder ejecutar el DVT dentro de la arquitectura de test, nos vemos obligados a convertir los instrumentos GPIB en instrumentos de red ethernet. Para solucionar este problema, es necesario emplear el gateway que es capaz de "convertir" los paquetes IP en comandos GPIB.

2.6.1. Interfaces

GPIB

El *Hewlett-Packard Instrument Bus* (HP-IB) fue un bus de datos desarrollado por Hewlett-Packard en los años 1970 para conectar dispositivos de test y medida (por ejemplo multímetros, osciloscopios, etc) con dispositivos que los controlen como por ejemplo un ordenador.

Más adelante, la Comisión Electrotécnica Internacional (IEC) y el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) deciden revisarlo y estandarizarlo bajo el nombre *Bus de Instrumentación de Propósito General* (GPIB).

Esto hizo que la inmensa mayoría de fabricantes de intrumentación lo adoptaran como un estándar de facto y por tanto se convirtiera en el bus común de conexión de instrumentos. A este bus se le conoce como IEEE 488.1 para diferenciarlo de versiones posteriores que introducen algunas mejoras.

Actualmente, la versión de GPIB integrada en los instrumentos es IEEE 488.2 que da soporte a los comandos SCPI. La versión cuenta con un rango de direccionamiento de 31 dispositivos que pueden ser conectados en cadena, en estrella o una combinación de ambas.

Figura 2.4: Conector GPIB

LXI

El estándar LXI (*LAN eXtensions for Instrumentation*) nace de la necesidad de conectar más dispositivos para aumentar las prestaciones y la velocidad de conexión. LXI es el sucesor de los estándares VXI (*VME eXtensions for Instrumentation*) y PXI (*PCI eXtensions for Instrumentation*) que tenían como propósito dar conectividad a dispositivos mediante tarjetas y bastidores. De esta manera lograban tener grandes dispositivos capaces de hacer múltiples medidas y en consecuencia se abandonarían los instrumentos de medidas concretas. Según esto, los diseñadores de estos buses pensaban que iba a suponer un ahorro para el usuario disponer de estos instrumentos y por tanto se abandonaría GPIB. La paradoja fue que era muchísimo más caro tener un sistema PXI o VXI capaz de realizar múltiples medidas que varios dispositivos capaces de funcionar "stand-alone".

Ante esta perspectiva, en 2004 surgió el estándar LXI que ofrece la posibilidad de conectar equipos independientes mediante ethernet, ofreciendo el más bajo coste de conectividad, inferior incluso que GPIB y a una velocidad más que suficiente para la gran mayoría de aplicaciones.

Los fabricantes de equipos de instrumentación pudieron migrar fácilmente los equipos GPIB a LXI sin necesidad de cambiar el formato de intrumento. El hecho de dar soporte a comandos SCPI hace que la migración sea transparente para el usuario y que por tanto no deba preocuparse del medio físico que transporta los comandos.

Al tratarse de dispositivos de red, se pueden direccionar tantos dispositivos como la red ethernet permita.

El bus LXI es un estándar desarrollado por el LXI Consortium, un consorcio de fabricantes de instrumentación que marca las especificaciones del bus, promueve el uso y asegura la interoperabilidad.

2.6.2. Arquitectura de comandos

SCPI

Los *Comandos Estándar para Instrumentación Programable* (SCPI) definen un estándar de comandos y sintaxis para ser usados en programación de dispositivos de medida.

En 1990, SCPI fue definido con las especificaciones del estándar IEEE 488.2. El estándar especifica una sintaxis común, una estructura de comandos y un formatos de datos para ser usado con todos los instrumentos que lo implementen. Fueron introducidos unos comandos genéricos (como son CONFigure y MEASure) que deben ser usados en cualquier instrumento. Los comandos están agrupados en subsistemas.

Como ya se ha comentado antes, este estándar define una capa de abstracción para el usuario y por tanto independientemente de la capa física empleada, el usuario envía el mismo formato de instrucción.

Un ejemplo de comando SCPI sería: **MEASure:VOLTage:DC?**

Arquitectura VISA

La arquitectura VISA (*Virtual Instrument Software Architecture*) es un estándar para la configuración y programación de instrumentos de medida que dispongan de los siguiente interfaces: GPIB, VXI, PXI, serie, ethernet, y/o interfaz USB. VISA es implementado por varias empresas del consorcio T&M (*Test & Measurement*) como son Rohde & Schwarz, Agilent Technologies, Anritsu, Bustec, National Instruments, Tektronix and Kikusui.

VISA proporciona el interfaz de programación entre el hardware y los entornos de desarrollo como Labview, LabWindows/CVI y otros lenguajes de programación. El estándar VISA se sitúa como el medio de transporte de los comandos SCPI y proporciona unas funciones de escritura y lectura muy sencillas de utilizar[4].

VISA, por tanto, se establece como un API (*Application Programming Interface*) para la programación de dispositivos de medida del cual derivan los conocidos *wrappers* en los diferentes lenguajes de programación. Nosostros, para la programación de los diferentes instrumentos utilizamos el wrapper de Python conocido como **PyVISA**.

Un diagrama jerárquico simplificado que representa la arquitectura VISA es el siguiente:

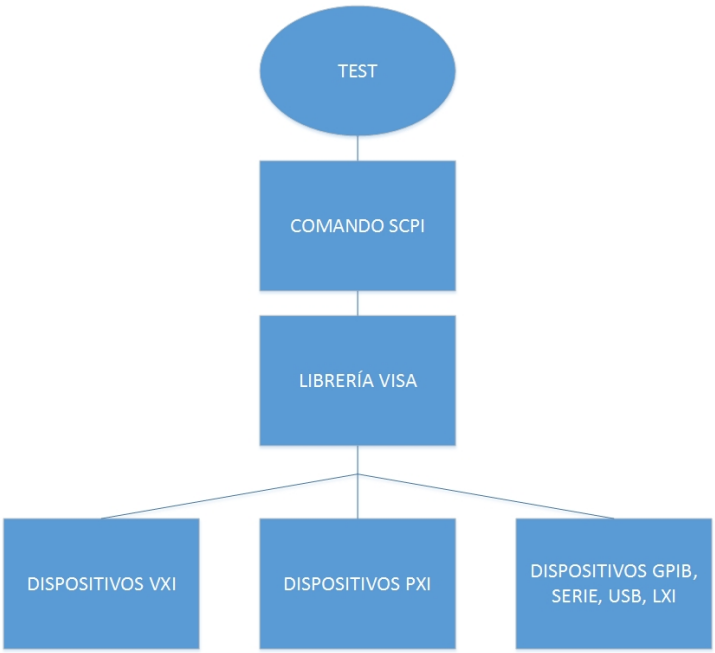


Figura 2.5: Arquitectura VISA

Capítulo 3

Descripción de la medida de *Puntos de Resposo*

3.1. Introducción

En un circuito electrónico, las tensiones contínuas nos aportan información acerca de la alimentación del mismo. Además de aportar información relativa a la alimentación del circuito, también nos aporta información del punto de equilibrio sobre el cual las señales alternas van a oscilar y por tanto saber cuál es la excursión máxima de la que disponemos.

Esta información es sumamente importante en circuitos basados en transistores ya que cualquier fallo de alimentación provoca una mala polarización del transistor y en consecuencia una excursión menor de la señal de interés. Esto se refleja directamente en un recorte de la señal a la salida.

La idea es poder medir la tensión continua de los pines de interés del circuito integrado y medir la corriente que consume el circuito integrado para tener controlada en todo momento la alimentación y el punto de equilibrio de los pines.

3.2. Descripción de la medida

La obtención de esta medida es muy sencilla por lo que respecta a la programación de la misma. Se trata de medir de una manera muy rápida la tensión continua en veinte puntos de medida (*Test Points*) y la corriente continua en otros dos puntos de medida.

Cabe destacar que la medida se ha de hacer en ausencia de señal a la entrada del DUT para medir correctamente valores continuos. De lo contrario mediríamos valores continuos más valores alternos y por tanto estaríamos falseando la medida.

Una vez obtenidos los valores, el test almacena dos ficheros de texto ASCII. Un fichero con extensión .meas que almacena de forma tabulada los valores obtenidos y otro fichero .setup que almacena con qué parámetros ha sido configurado el dispositivo.

Esta medida tan rápida es posible gracias al **data logger** que dipone de una tarjeta de adquisición compuesta por veinte canales para medida de tensión (CH1-CH20) y dos canales para medida de corriente (CH21,CH22).

El código que forma la medida automática es:

Recopila información de configuración
Abre sesión data logger
Obtiene VDC CH1-CH20
Obtiene IDC CH21,CH22
Cierra sesión data logger
Tabula y almacena fichero .meas con las medidas obtenidas
Tabula y almacena fichero .setup con los parámetros de configuración del dispositivo
Devuelve vector con información organizada para la realización del informe final al finalizar todas las iteraciones

3.3. Descripción general de ajustes del data logger

Tal y como hemos comentado antes, el data logger es el instrumento que nos proporciona la capacidad de automatizar esta medida. La configuración del data logger para este fin es gracias a la tarjeta de adquisición que permite realizar dichas medidas.

Cabe destacar que el data logger dispone de tres bahías donde se pueden introducir diversas tarjetas de adquisición con diversas funcionalidades tales como canales de medida independientes, canales con masa común... así como tarjeta de relés entre otras.

La tarjeta de adquisición y el esquema eléctrico de la misma se puede observar en la figura 3.1

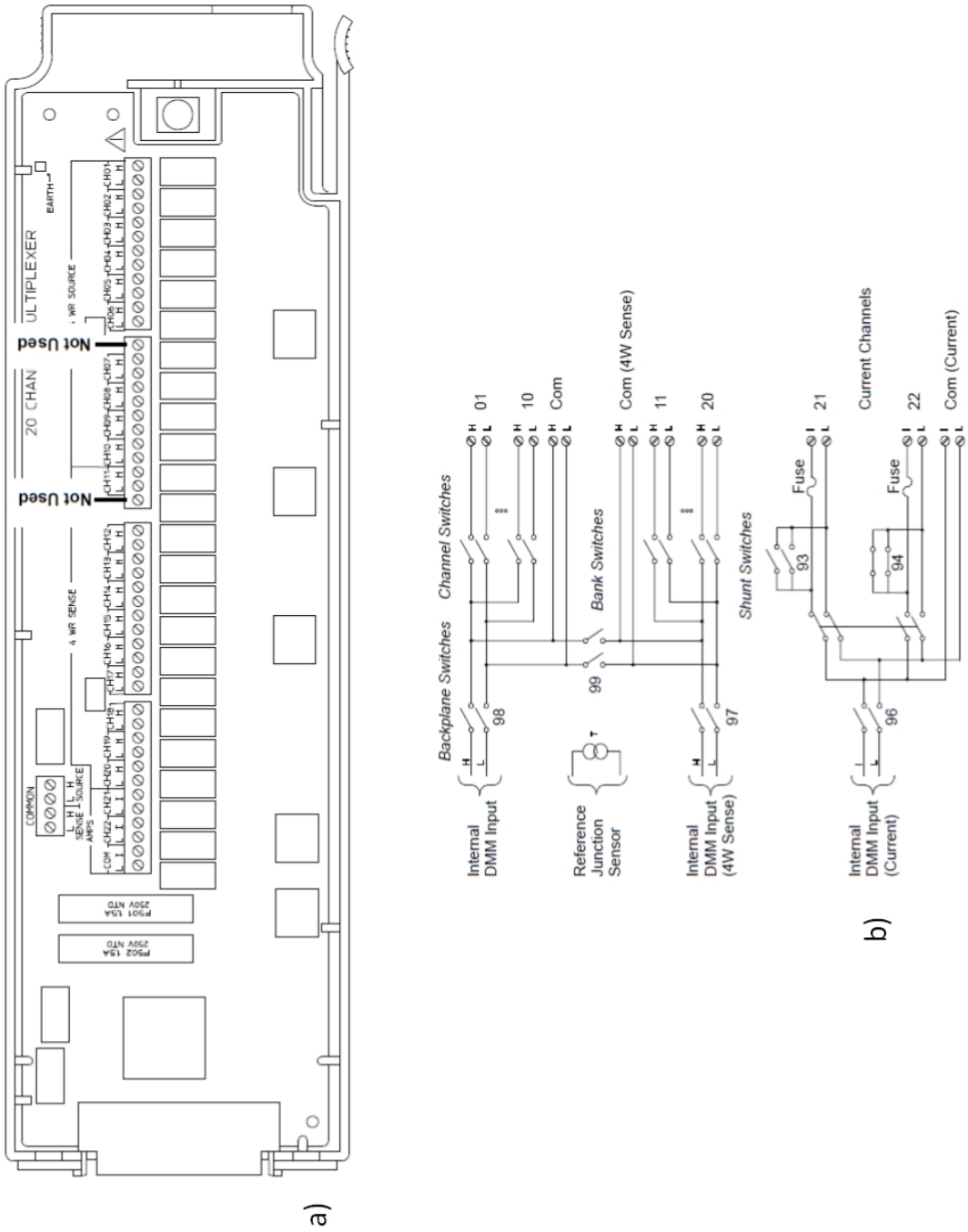


Figura 3.1: a) Tarjeta de adquisición y b) esquema eléctrico

3.4. Ficheros generados

A continuación, se puede observar una ejemplo del fichero de medidas (.meas) donde la información se encuentra tabulada por canal de adquisición, es decir, la primera columna muestra el número del canal y la seguna columna el valor medido en voltios (CH1-CH20) o amperios (CH21,CH22):

Vdc1	*+X.63228740E+00*
Vdc2	*+X.63214930E+00*
Vdc3	*+X.62662560E+00*
Vdc4	*-X.84100000E-05*
Vdc5	*+X.62727360E+00*
Vdc6	*+X.61064940E+00*
Vdc7	*+X.61054320E+00*
Vdc8	*+X.60261880E+00*
Vdc9	*+X.65797260E+00*
Vdc10	*+X.51646340E+00*
Vdc11	*+X.51642090E+00*
Vdc12	*+X.51634650E+00*
Vdc13	*+X.51638900E+00*
Vdc14	*+X.62404440E+00*
Vdc15	*+X.62719920E+00*
Vdc16	*+X.10870930E-02*
Vdc17	*+X.17185050E-02*
Vdc18	*+X.16554060E-02*
Vdc19	*+X.12947660E-02*
Vdc20	*+X.18302920E+00*
Idc21	*-X.18709930E-02*
Idc22	*+X.58229330E-03*

En el siguiente ejemplo se puede observar el fichero que almacena la configuración del instrumento (.setup):

```
DATA LOGGER: "XXXX,XXXXXX,MY49012080,1.15-1.12-02-02"
CONF_1: VOLTAGE=INPUT:IMPEDANCE:AUTO ON,(@101:120);MEAS:VOLT:DC? (@101:120)
CONF_2: CURRENT=MEAS:CURR:DC? (@121:122)
SPI_FRAME = XXXXXXXX_XXXXXXXX_XXXXXXXX_XXXXXXXX
```

Capítulo 4

Descripción de la medida *Densidad Espectral de Ruido*

4.1. Introducción

Cuantificar el ruido es uno de los factores más importantes en telecomunicaciones y nos proporciona información acerca de la calidad de la comunicación. El hecho de tener niveles muy altos de ruido provoca que esta relación empeore y en consecuencia perder calidad en la comunicación.

Es muy importante poder cuantificar la potencia de ruido intrínseco que introduce un sistema electrónico para analizar la calidad de la comunicación.

Claro está que un sistema electrónico amplificador no realiza la amplificación de la señal sin ningún tipo de perjuicio, por tanto, tan importante es garantizar unos buenos niveles de amplificación como garantizar que los niveles de ruido no son excesivamente altos.

4.2. Descripción de la medida

La misión de esta medida es calcular y representar la densidad espectral de ruido en la región de interés.

La obtención de la medida se realiza gracias al **analizador de espectro** y en ausencia de señal a la entrada del DUT para poder verificar el ruido intrínseco del circuito integrado analógico.

Siguiendo la dinámica de tener en todo momento la información almacenada sin ambigüedad, la medida nos proporciona tres ficheros ASCII con toda la información. Un fichero de texto .meas que almacena la información tabulada compuesta por todas las parejas de puntos (Frecuencia, DEP ruido), un fichero de texto .setup que almacena la configuración del dispositivo de medida (analizador de espectro) y un fichero gráfico (.png) que almacena la representación de la DEP de ruido.

El pseudocódigo de la medida es el siguiente:

Recopila información de configuración
Abre sesión analizador de espectro
Establece configuración del analizador de espectro (atenuador, frecuencia inicial, frecuencia final...)
Captura la traza y aplica el factor de correción para tener valores de DEP
Cierra sesión analizador de espectro
Genera el fichero .png con la DEP
Tabula y almacena fichero .meas con las medidas obtenidas
Tabula y almacena fichero .setup con los parámetros de configuración del dispositivo
Devuelve vector con información organizada para la realización del informe final al finalizar todas las iteraciones

4.3. Descripción general de ajustes del analizador de espectro

El ruido es un proceso estocástico formado por un conjunto de variables aleatorias que evolucionan en función de otra variable, generalmente el tiempo.

Al tratarse de un conjunto de variables aleatorias, la forma natural de tratar el ruido es mediante el valor cuadrático medio de la señal (RMS). El valor cuadrático medio nos proporciona el nivel de potencia de ruido que en esencia es la información de interés. Los valores instantáneos no aportan información relevante por ser valores aleatorios.

En consecuencia, para medir la DEP de ruido el analizador de espectro ha de ser configurado con el detector **RMS** para obtener valores de potencia de ruido, además de los ajustes de frecuencias, atenuador, filtro de resolución y filtro de vídeo.

Cabe destacar que el ajuste del filtro de resolución fija el tiempo de barrido. Por tanto la elección del ajuste del filtro de resolución se realiza de manera que nos proporcione una traza con suficiente resolución pero que no dispare excesivamente el tiempo de barrido.

4.4. Descripción del factor de corrección

El analizador de espectro durante el barrido realiza una discretización en frecuencia. Para cada punto de frecuencia, el analizador de espectro calcula el valor cuadrático medio (porque el detector es RMS) de la señal que pasa por el ancho de banda filtro de resolución, entonces, para cada punto de frecuencia tenemos un valor de potencia de ruido calculado.

Si cada uno de estos puntos lo dividimos por el ancho de banda del filtro de resolución obtenemos la DEP de cada punto. Como nos encontramos trabajando en escala logarítmica esta corrección se convierte en una resta.

Por tanto, una vez obtenido el vector de puntos que forman la traza, el vector es corregido mediante el factor de correción para obtener la traza de densidad espectral de potencia. Entonces la operación queda de la siguiente manera:

$$DEP[i](dBm/Hz) = (Vector\ Potencia_{ruido}[i](dBm)) - 10log(BW_{filtro\ resol}) \tag{4.1}$$

En la figura 4.1 se puede ver como se calcula la DEP en un analizador de espectro.

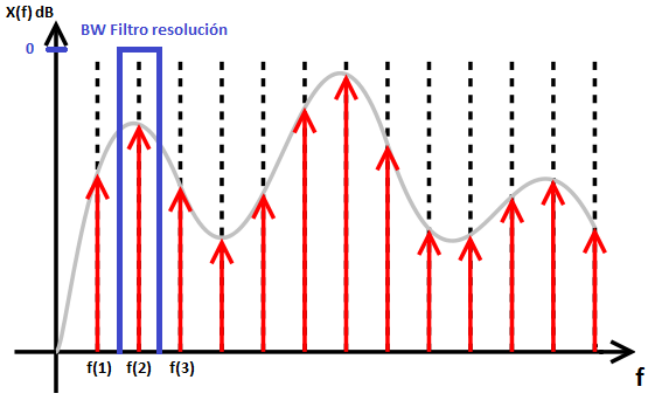


Figura 4.1: Representación DEP en un analizador de espectro

4.5. Ficheros generados

A continuación, se muestra un ejemplo del fichero de medidas (.meas) donde cada pareja de puntos representa la frecuncia en kHz y el valor de DEP en dBm/Hz:

```
X000.0              -X39.81492615
X395.209              -X40.1260757
X790.418              -X40.166542
X185.627              -X39.68785858
X580.836              -X41.1613006
X976.045              -X39.64621735
X371.254              -X41.0074157
X766.463              -X40.8400573
X161.672              -X40.8302917
X556.881              -X40.7588958
...
```

El siguiente ejemplo muestra el fichero de configuración (.setup):

```
NOISE PSD measurement
SPECTRUM ANALYZER CONFIGURATION
"XXXX,XXXXX,100017/008,4.25"
ATT = X
REF_LEVEL = -X0
START_FREQUENCY = X000000
STOP_FREQUENCY = X00000000
RESOL_BW = X0000
VIDEO_BW = X00000
REF_LEVEL_OFFSET = X
DETECTOR = RMS
SPI_FRAME = XXXXXXXX_XXXXXXXX_XXXXXXXX_XXXXXXXX
```

Por último, se muestra un ejemplo en la figura 4.2 de una DEP dibujada por la medida automática. Se ha optado por una representación semilogarítmica para poder ver mejor el efecto de los ceros y polos.

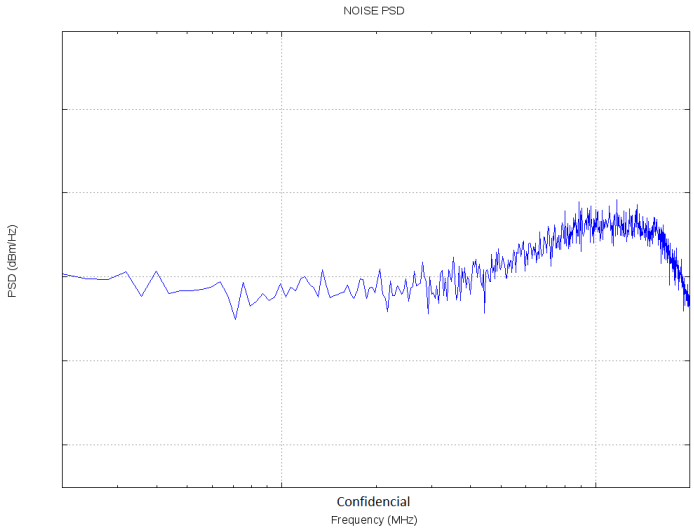


Figura 4.2: Gráfica de una DEP de ruido

Capítulo 5

Descripción de la medida
Función de Transferencia

5.1. Introducción

En un sistema electrónico de comunicaciones, la función de transferencia es con toda probabilidad la medida de mayor interés porque nos proporciona la respuesta del sistema para una entrada determinada.

Un sistema electrónico presenta una respuesta impulsional única siempre que no cambien las condiciones del sistema. Esto quiere decir que para un sistema electrónico, mientras no entren en juego las derivas de los componentes, la respuesta impulsional será la misma y por tanto se trata de una respuesta identificativa de este sistema.

Esta respuesta impulsional o función de transferencia se puede obtener matemáticamente en el dominio del tiempo o en el dominio frecuencial. Ambas respuestas son equivalentes pero para la obtención física la manera práctica y factible de hacerlo es en el dominio frecuencial.

En el dominio temporal la forma de obtener la función de transferencia es mediante la siguiente operación:

$$y(t) = x(t) * h(t) \tag{5.1}$$

siendo $x(t)$ la excitación a la entrada e $y(t)$ la respuesta a la salida. Por tanto, $y(t) = h(t)$ porque el estímulo de entrada es una delta de Dirac ($x(t) = \delta(t)$).

La dificultad de obtener una delta de Dirac[1] en el mundo físico nos lleva a tener que trabajar en el dominio frecuencial.

En el dominio frecuencial, podemos obtener la respuesta equivalente mediante la siguiente operación:

$$H(\omega) = \frac{Y(\omega)}{X(\omega)} \tag{5.2}$$

siendo $X(\omega) = TF[x(t)]$ e $Y(\omega) = TF[y(t)]$.

Como $\omega = 2\pi f$ podemos trabajar directamente con la siguiente expresión:

$$H(f) = \frac{Y(f)}{X(f)} \tag{5.3}$$

En términos de potencia tenemos:

$$|H(f)|^2 = \frac{|Y(f)|^2}{|X(f)|^2} \tag{5.4}$$

5.2. Descripción de la medida

El objetivo de esta medida es calcular y representar la función de transferencia en la región de interés.

[1] No hay que perder de vista que la delta de Dirac es una herramienta matemática imposible de reproducir físicamente porque se define como $\delta(t = 0) = \text{ínf}$, $\delta(t \neq 0) = 0$

La obtención de la medida se realiza mediante el **analizador de espectro** por el hecho de estar trabajando en el dominio frecuencial. A diferencia de medidas anteriores, en ésta, si que es necesario excitar el DUT con una señal determinada. Esta señal, como hemos explicado antes, ha de ser una señal concreta que nos permita obtener la función de transferencia del circuito. En principio, trabajando en el dominio frecuencial, cualquier señal sería válida, pero para simplificar los cálculos emplearemos una señal plana en frecuencia.

Para inyectar esta señal vamos a hacer uso del **generador de señales arbitrarias**, que inyectará una señal chirp[2].

Como en medidas anteriores, esta medida nos proporciona tres ficheros. El fichero de configuración (.setup), el fichero de medidas (.meas) y el gráfico con la representación de la función de transferencia (.png).

El pseudocódigo que ejecuta la medida es el siguiente:

[2]Una señal chirp es un tono que cambia la frecuencia en función del tiempo de forma lineal

Recopila información de configuración
Abre sesión generador de señales
Carga señal chirp y establece configuración del generador de señales (v_{out}, frecuencia de muestreo)
Abre sesión del analizador de espectro
Establece configuración del analizador de espectro (atenuador, frecuencia inicial, frecuencia final...)
Activa la salida del generador de señales
Captura la traza y aplica el factor de correción para tener valores de función de transferencia
Cierra sesión generador de señales
Cierra sesión analizador de espectro
Genera el fichero .png con la función de transferencia
Tabula y almacena fichero .meas con las medidas obtenidas
Tabula y almacena fichero .setup con los parámetros de configuración del dispositivo
Devuelve vector con información organizada para la realización del informe final al finalizar todas las iteraciones

5.3. Descripción general de ajustes del analizador de espectro

Para esta medida el analizador de espectro se configura con los ajustes que nos ofrecen una buena resolución sin alargar mucho el tiempo de barrido. El detector se establece como detector de **pico positivo** porque se trata de una señal plana en frecuencia y así evitamos capturar picos negativos espúreos.

5.4. Descripción general de ajustes del generador de señales arbitrarias

El generador carga la señal chirp[3] y establece una v_{out} y una **frecuencia de muestreo**. La frecuencia de muestreo se establece al máximo para tener una señal chirp suficientemente rápida como para poder ver una señal plana en el analizador con los ajustes establecidos. La v_{out} ha de variar en función de qué nivel de ganancia[4] está establecida en el DUT para verificar que el dispositivo atenua y amplifica y no saturarlo.

5.5. Descripción del factor de correción

Al igual que en la medida anterior, es habitual trabajar con valores logarítmicos. Por tanto, la expresión

$$|H(f)|^2 = \frac{|Y(f)|^2}{|X(f)|^2} \tag{5.5}$$

se convierte en términos de frecuencia en $|H(f)|^2(dB) = |Y(f)|^2(dBm) - |X(f)|^2(dBm)$.

Los valores del factor de correción $|X(f)|^2(dBm)$ han sido previamente calculados con el analizador de espectro. Las ganancias del DUT han sido agrupadas en grupos de 3 o 4. Para cada uno de estos grupos, el DUT ha sido excitado con la misma señal chirp pero con diferente potencia, entonces cada grupo se ha corregido con su factor de corrección.

[3]Señal generada mediante software como Matlab, Octave... y almacenada en el generador
[4]El DUT puede tener ganancias positivas y negativas

5.6. Ficheros generados

En esta medida, como en la anterior, se generan tres ficheros. A continuación se puede ver un extracto del fichero de medidas (.meas) donde la primera columna corresponde a frecuencias (kHz) y la segunda columna corresponde a valores de función de transferencia (dB):

```
X000.0              -X7.19024658
X395.209              -X7.25233078
X790.418              -X7.06286621
X185.627              -X7.12130737
X580.836              -X7.17953491
X976.045              -X7.2187767
X371.254              -X7.19576263
X766.463              -X7.14412689
X161.672              -X7.43893051
X556.881              -X7.14184952
...
```

Un ejemplo del fichero de configuración (.setup):

```
TRANSFER FUNCTION
ARBITRARY WAVEFORM GENERATOR
"XXXX,XXXXXXXX,B010147,SCPI:99.0 FW:4.5.0.6 "
SIGNAL = Z:/signals/chirp/XXXXXXX.awg
OUTPUT VOLTAGE = X
SAMPLING RATE = X200000000
SPECTRUM ANALYZER CONFIGURATION
"XXXX,XXXXX,100017/008,4.25"
ATT = X0
REF_LEVEL = X
START_FREQUENCY = X000000
STOP_FREQUENCY = X00000000
RESOL_BW = X0000
VIDEO_BW = X0000
```

```
REF_LEVEL_OFFSET = X
DETECTOR = POSITIVE
SPI_FRAME = XXXXXXXX_XXXXXXXX_XXXXXXXX_XXXXXXXX
```

Por último, la figura 5.1 muestra un ejemplo de la traza de la función de transferencia almacenada en el fichero (.png):

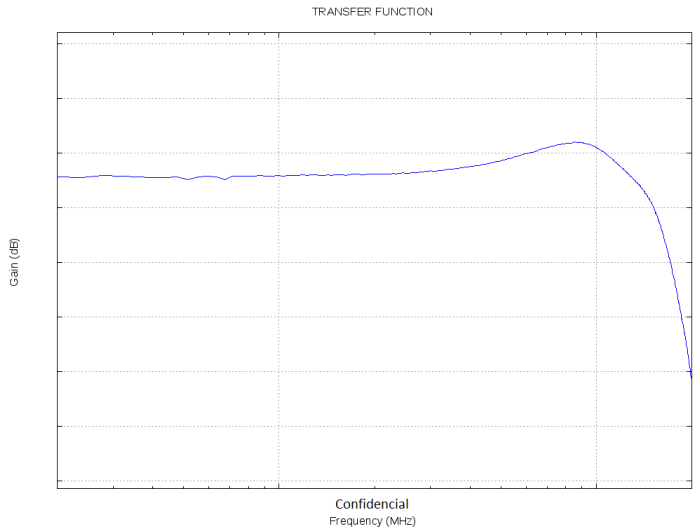


Figura 5.1: Gráfica de una función de transferencia

Capítulo 6

Descripción de la medida de *Consumo dinámico*

6.1. Introducción

Medir el consumo eléctrico del DUT en funcionamiento dinámico nos puede ayudar a observar anomalías en el funcionamiento. Cuando hablamos de modo dinámico nos referimos al funcionamiento del DUT con señales OFDM con las cuales va a trabajar habitualmente.

Esta medida es prácticamente igual a la medida de puntos de reposo con la salvedad que únicamente nos interesa el consumo de corriente que realiza el DUT cuando lo estamos excitando con señales OFDM.

6.2. Descripción de la medida

El objetivo es cubrir la necesidad de saber cuánto consume el DUT en modo de funcionamiento normal.

Para ello ejecutamos el siguiente código:

Recopila información de configuración
Abre sesión generador de señales
Carga señal OFDM y establece configuración del generador de señales (v_{out}, frecuencia de muestreo)
Activa la salida del generador de señales
Abre sesión del data logger
Obtiene IDC CH21,CH22
Cierra sesión data logger
Cierra sesión generador de señales
Tabula y almacena fichero .meas con las medidas obtenidas
Tabula y almacena fichero .setup con los parámetros de configuración del dispositivo
Devuelve vector con información organizada para la realización del informe final al finalizar todas las iteraciones

6.3. Descripción general de ajustes del data logger

En esta sección aplican los mismos conceptos descritos en el apartado 3.3

6.4. Descripción general de ajustes del generador de señales arbitrarias

El generador de señales excita el DUT con una señal ODFM que tiene una frecuencia de muestro determinada (frecuencia de muestreo a la cual se ha generado) y con una v_{out} concreta para cada ganancia sin que sature el DUT.

6.5. Ficheros generados

Al igual que en la medida de puntos de resposo, esta medida sólo genera dos ficheros. El fichero de medidas (.meas) del que podemos ver un ejemplo:

```
Idc21                -X.18624780E-02
Idc22                +X.57871920E-03
```

y el fichero de ajustes (.setup):

```
DATA LOGGER: "XXXX,XXXXXX,MY49012080,1.15-1.12-02-02"
CONF_2: CURRENT=MEAS:CURR:DC? (@121:122)
SPI_FRAME = XXXXXXXX_XXXXXXXX_XXXXXXXX_XXXXXXXX
```

Capítulo 7

Descripción de la medida de *MultiTone Power Ratio*

7.1. Introducción

En los capítulos anteriores nos hemos centrado en intentar caracterizar el consumo, el ruido y la función de transferencia del DUT. En esta medida, el objetivo es caracterizar la **no** linealidad del DUT que hasta ahora no habíamos tenido en cuenta.

La medida de MTPR se realiza mediante una señal OFDM que consiste en una serie de portadoras discretizadas en frecuencia en una región determinada.

A esta señal se le suprimen una serie de portadoras (notches) y se observa el nivel de potencia que existe en estos notches al atravesar el sistema. La relación entre el nivel de portadores y el nivel de potencia del notch es el valor de MTPR en la frecuencia del notch.

Esta relación a lo largo de todos los notches, genera la figura de MTPR que nos da una visión de las no linealidades del DUT a lo largo de la región de interés.

7.2. Descripción de la medida

El código que forma la MTPR es el siguiente:

Recopila información de configuración
Abre sesión generador de señales
Carga señal OFDM y establece configuración del generador de señales (v_{out}, frecuencia de muestreo)
Abre sesión del analizador de espectro
Establece configuración del analizador de espectro (atenuador, frecuencia inicial, frecuencia final...)
Activa la salida del generador de señales
Calcula la separación entre portadoras
Calcula la separación entre notches (num. portadoras · separación portadoras)
for NOTCH **do**
    Frecuencia central = frecuencia notch
    Captura la traza
    Calcula el valor de MTPR
    Almacena valores en un vector
end for
Cierra sesión generador de señales
Cierra sesión analizador de espectro
Genera el fichero .png con la representación de la MTPR
Tabula y almacena fichero .meas con las medidas obtenidas
Tabula y almacena fichero .setup con los parámetros de configuración del dispositivo
Devuelve vector con información organizada para la realización del informe final al finalizar todas las iteraciones

En la figura 7.1 se muestra el concepto de MultiTone Power Ratio:

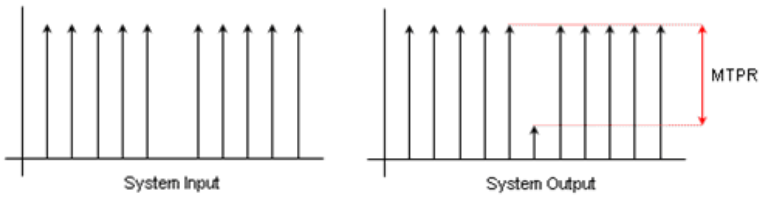


Figura 7.1: MultiTone Power Ratio

A partir de la traza de la figura 7.2 obtenemos la separación entre portadoras. Por tanto, conociendo el número de portadoras y la separación calculada podemos centrarnos en los diferentes notches (figura 7.3) e ir obteniendo el valor de MTPR en la frecuencia del notch.

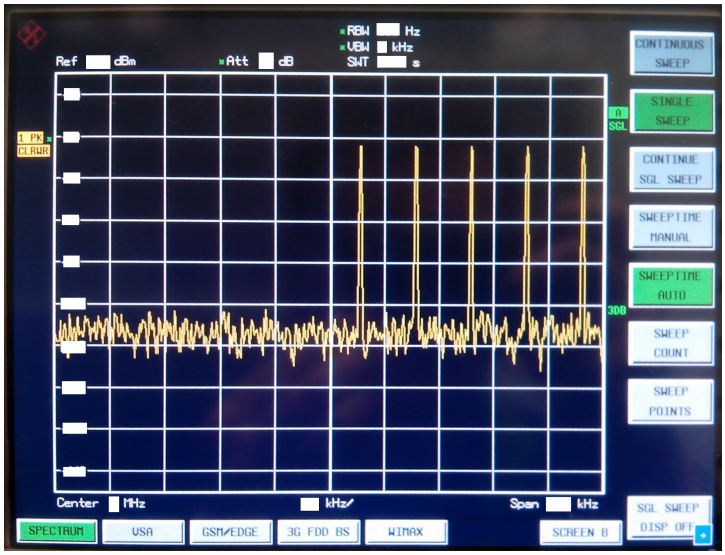


Figura 7.2: Separación entre portadoras

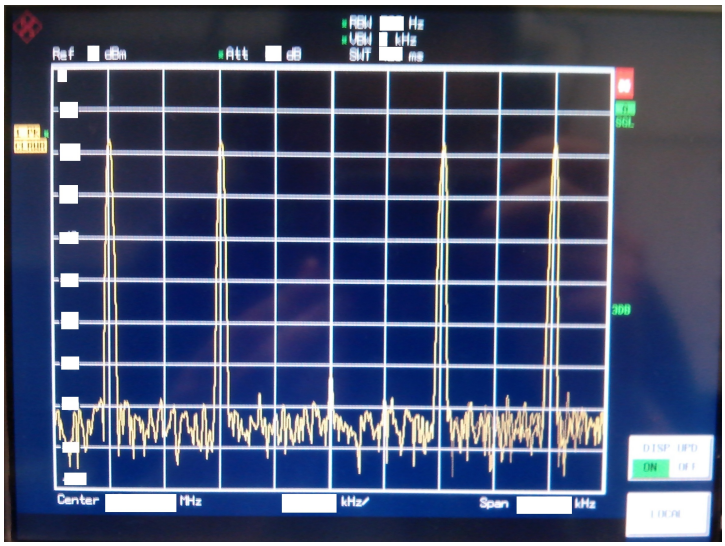


Figura 7.3: Notch MTPR

7.3. Descripción general de ajustes del analizador de espectro

En cuanto a los ajustes del analizador de espectro para la MTPR, no hay cambios significativos respecto a la medida de la función de transferencia. Se usan unos ajustes con suficiente resolución pero que no ralentice demasiado el tiempo de barrido. Cabe destacar que para la medida de la separación entre portadoras se usa un filtro de resolución menor que en el barrido del notch para tener mejor resolución en el cálculo de la separación.

7.4. Descripción general de ajustes del generador de señales arbitrarias

Para la medida de la MTPR se emplea una señal OFDM con su frecuencia de muestreo y una v_{out} que no sature.

7.5. Ficheros generados

Como en cualquiera de las medidas anteriores, se almacena toda la información posible para tenerla accesible de manera sencilla. Podemos ver un ejemplo del fichero de medidas (.meas) siendo la primera columna frecuencia del notch (kHz) y la seguna columna el valor de MTPR (dBc):

```
X315.45337425          X8.75104714
X591.30708683          X0.40975952
X965.65645808          X4.37676049
X290.65645808          X6.48748016
X615.65645808          X4.40714454
X904.10136826          X8.40994835
X265.45337425          X6.41123199
X590.65645808          X3.93295288
X915.45337425          X1.95617103
X240.65645808          X7.31936264
...
```

El siguiente ejemplo muestra la estructura del fichero de configuración (.setup):

```
MTPR
ARBITRARY WAVEFORM GENERATOR
"XXXX,XXXXXXXX,B010147,SCPI:99.0 FW:4.5.0.6 "
SIGNAL = Z:/signals/OFDM/XXXXXXXX/X_XX_XXXXXX_XXX_XXXXX.awg
OUTPUT VOLTAGE = X
SAMPLING RATE = X00000000
SPECTRUM ANALYZER CONFIGURATION
"XXXX,XXXXX,100017/008,4.25"
ATT = X0
REF_LEVEL = X
RESOL_BW = X00
VIDEO_BW = X000
REF_LEVEL_OFFSET = X
DETECTOR = POSITIVE
SPI_FRAME = XXXXXXXX_XXXXXXXX_XXXXXXXX_XXXXXXXX
```

Por último se muestra un gráfico de una MTPR:

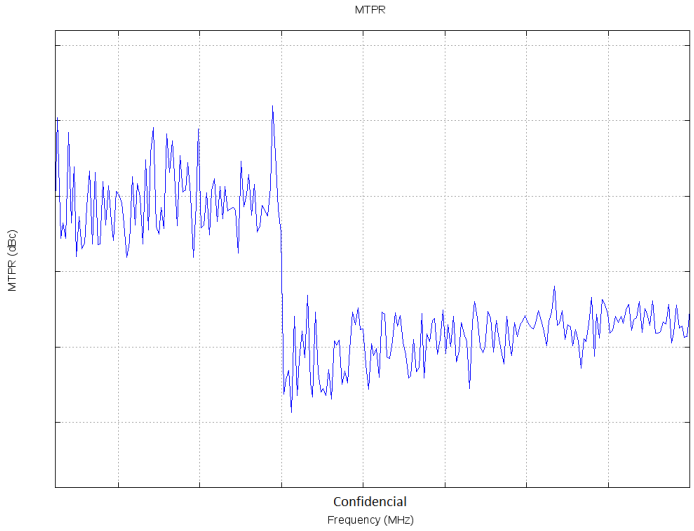


Figura 7.4: Ejemplo MTPR

Capítulo 8

Módulo SPI máster

8.1. Introducción

El DUT es un sistema amplificador de ganancia configurable. Para realizar dicha configuración, el sistema incluye un interfaz serie de comunicación basado en el protocolo SPI (*Serial Peripheral Interface*) capaz de recibir tramas de 32 bits.

8.2. Bus SPI

El Bus SPI es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos.

El bus SPI está compuesto por un interfaz de cuatro señales:

- Chip Select (CS)
- Master Input Slave Output (MISO)
- Master Output Slave Input (MOSI)
- Serial clock (CLK)

Este interfaz tiene la posibilidad de trabajar en cuatro modos de funcionamiento. Los modos de funcionamiento están relacionados con la polaridad y con la fase del reloj. Una buena ilustración de este concepto se muestran en la figura 8.1 donde podemos observar la cuatro combinaciones posibles[10].

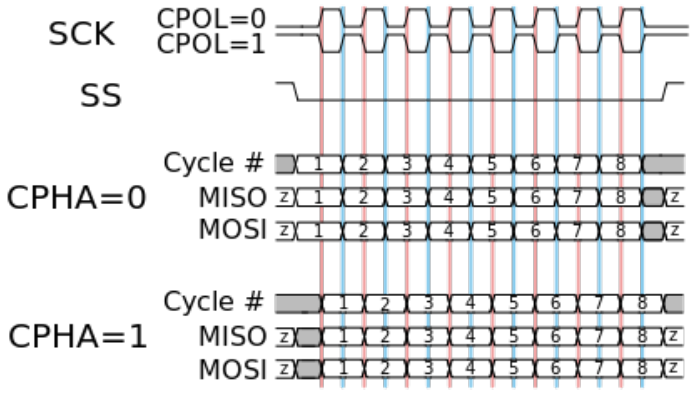


Figura 8.1: Modos de funcionamiento SPI

Nuestro DUT emplea CPOL=1 y CPHA=1 que es conocido como SPI_MODE=3.

8.3. Módulo SPI máster

Para configurar el DUT, es necesario usar algún dispositivo que desempeñe el rol de máster para poder configurar nuestro DUT (esclavo). Para alcanzar este objetivo optamos por emplear un microcontrolador que fuese capaz de transmitir tramas SPI de 32 bits.

Entre muchos microcontroladores nos decantamos por usar un módulo arduino. Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares. En particular, usamos el módulo arduino leonardo que está basado en el Atmega32u4.

Los módulos arduino disponen de un bootloader que se encarga de la programación del microcontrolador mediante el puerto USB-serie[1] para facilitar el proceso. De no ser así, sería obligado usar un programador externo para programar el microcontrolador. Con el bootloader, durante los primeros segundos después de un reset, se habilita un puerto USB-serie auxiliar mediante el cual se realiza la carga del programa. Pasados estos segundos se deshabilita este puerto y se habilita el puerto USB-serie de entrada/salida.

El objetivo de este módulo SPI máster es retransmitir por el bus SPI las tramas de 32 bits que recibe por el puerto serie. La figura 8.2 muestra el módulo empleado (ATMEGA32U4 Breakout DEV-11117 de sparkfun) que es equivalente al arduino leonardo.

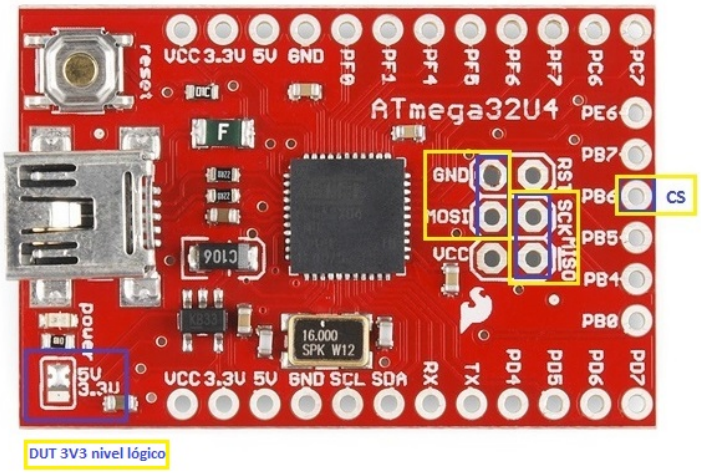


Figura 8.2: Módulo SPI máster

[1] Un puerto USB-serie es un puerto físico USB que funciona como un puerto serie. Esto quiere decir que la conexión física se realiza mediante un puerto USB pero en el PC es reconocido como un puerto serie. También es conocido como *Virtual COM Port*

8.4. Código programación módulo SPI máster

Como se ha comentado antes, la tarea en sí del módulo es retransmitir por el bus SPI las tramas de 32 bits que reciba por el puerto USB-serie. Para ello el algoritmo que realiza la función es el siguiente:

```
Define variables
Configura pin de CS como output
Inicia el módulo SPI
Establece el orden MSB primero
Establece el modo de funcionamiento 3
Establece la frecuencia de reloj
Establece el pin de CS a nivel alto
loop
    if Puerto serie ≥ 4bytes then
        Lee puerto serie y almaceno el contenido en una variable
        Establece pin CS a nivel bajo
        Retransmite el contenido de la variable por MOSI y lee/almacena el
contenido de MISO (se realiza a la par en el mismo flanco de reloj)
        Establece pin CS a nivel alto (fin de la transmisión)
        Retransmite el contenido leído de MISO por USB-serie al PC
        Cierra puerto serie
    end if
end loop
```

8.5. Código de retransmisión por puerto serie

Para enviar por USB-serie las tramas al módulo SPI, empleamos un script en Python que se ejecuta en el PC auxiliar. Este script es instalado en un directorio de trabajo específico y se ejecutan llamadas al script por ssh desde el PC lanzador. Mediante la llamada al script, se pasan como argumento las tramas de 32 bits

en hexadecimal y este se encarga de retransmitir y recibir las tramas del puerto USB-serie. El algoritmo del script es el siguiente:

```
Define variables
Almacena en un vector los argumentos (tramas TX en hexadecimal)
Convierte de hexadecimal a binario
if Puerto serie existe then
    for length(vector tramas TX) do
        Retransmite trama por puerto serie
    end for
    for length(vector tramas TX) do
        Lee tramas de puerto serie y convierte a hexadecimal
        Almacena en un vector tramas RX
    end for
else
    Excepción
end if
Print vector tramas RX
```

Capítulo 9

Interfaz gráfico de usuario

9.1. Introducción

En la introducción del presente proyecto se ha comentado la implementación de una GUI para lanzar de una manera simple e individual casos de test que sean de nuestro interés.

Esta GUI pretende ser un interfaz gráfico que genere de manera sencilla el fichero de configuración del DVT y haga una llamada a Robot Framework para la ejecución del mismo.

9.2. Descripción de la GUI

La GUI se ha planteado como un interfaz donde se ven representados los diferentes instrumentos que intervienen en un test y mediante la cual enviamos la configuración deseada a cada uno de ellos. Una vez establecidos los ajustes, se indican que tests nos interesan ejecutar.

La interfaz ha sido dividida en bloques organizados[1] de izquierda a derecha y de arriba a bajo para conducir al usuario durante su uso. La figura 9.1 muestra la ventana de la GUI.

Para la implementación de la GUI nos decantamos por el uso del RAD (*Rapid Application Development*) Glade que nos facilita el proceso de diseño e implementación. Glade nos permite desarrollar un interfaz gráfico de manera visual mediante la tecnología *drag and drop* donde se disponen los elementos mediante contenedores.

Glade utiliza la librería GTK (*GNOME ToolKit*) para la generación del fichero XML (fichero ASCII). La librería GTK es independiente del lenguaje de programación empleado y mediante wrappers se extiende a los diferentes lenguajes de programación. En este caso particular, vamos a emplear el wrapper **PyGTK** para la programación en Python.

De manera más esquemática, la implementación de la GUI se puede resumir en los siguientes pasos:

1. Diseño gráfico de la ventana mediante Glade
2. Glade genera un fichero ASCII que contiene etiquetas xml y GTK para realizar la conexión con cualquiera de los wrappers
3. Se importa el fichero ASCII al fichero de programación python
4. Se conectan los eventos con la función asociada a cada evento
5. Se programan las funciones como si de cualquier programa se tratara

De esta manera, es muy sencillo y muy intuitivo la creación de cualquier GUI.

Hay que destacar que para la ejecución de tests con la GUI hemos tenido que modificar el fichero DVT.txt. Esta modificación es debido a que el DVT lleva consigo una construcción iterativa para ejecutarse en el rango de temperaturas,

[1] Aunque están organizados, son bloques independientes pero se recomienda el uso secuencial al menos la primera vez

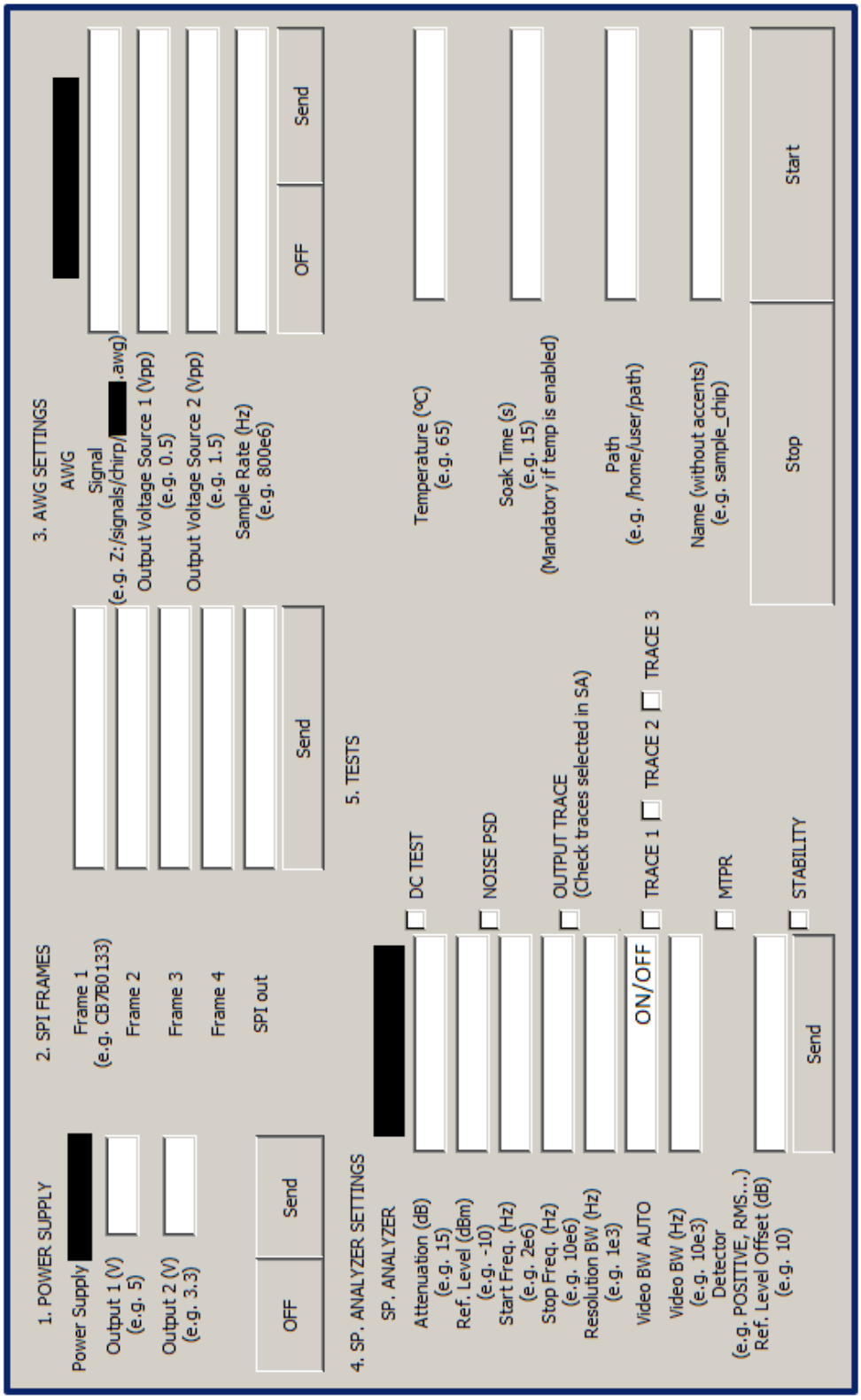


Figura 9.1: GUI

tensiones y ganancias especificado. Como nuestra GUI se ha implementado con la finalidad de realizar ejecuciones particulares, no tiene sentido hacer llamadas al fichero DVT.txt que requiere de muchos más argumentos de entrada.

Para conseguir esto hemos hecho una copia del fichero DVT.txt sobre la cual se han realizado los pequeños cambios.

9.3. Ficheros generados

Una vez establecidos los ajustes de los instrumentos mediante los distintos bloques de la GUI procederíamos a marcar qué test son de nuestro interés y a qué temperatura lo deseamos pasar. Con esta información se genera un pequeño fichero de configuración que sirve como argumentos de entrada al programa principal (GUI-DVT.txt modificado anteriormente).

Un ejemplo del fichero de configuración (.py) generado por la GUI es el siguiente:

```
ENABLE_TEMPERATURE = 0
TEST2RUN = [False,True,False,False,False]
ENABLED_TRACES = [False,True,False]
FILENAME = "Prueba"
SOAK = "5"
```

Capítulo 10

Conclusiones y líneas futuras

10.1. Conclusiones

En la introducción de este proyecto se ha comentado la importancia que tiene realizar una buena validación de un diseño. Poder realizar una buena validación e independizar el proceso de una persona supone grandes ahorros en términos temporales.

Este DVT ha sido diseñado en torno a un circuito integrado concreto. La doble finalidad de todo este trabajo ha sido la posible reutilización y expansión a futuros diseños del mismo propósito pero con mayores capacidades. Con esta perspectiva se ha intentado diseñar el DVT para que la adaptación del DVT a futuros diseños sea lo más fácil e intuitiva.

De una forma rápida, se ha evaluado el coste temporal de un determinado juego de casos de test mediante la presente herramienta y de forma manual. El resultando este sistema ha sido de ocho horas de ejecución mientras que de forma manual podría llevar días incluso semanas.

De esta manera, hemos podido comprobar el ahorro temporal que supone además de la libertad de trabajo para el personal durante este tiempo.

10.2. Líneas futuras

Una buena manera de continuar con este trabajo podría ser la introducción del test de estabilidad. Esta medida aporta información acerca de la estabilidad del DUT.

Se ha realizado un estudio para la implementación de esta medida pero debido a inconvenientes del dispositivo de medida (osciloscopio digital) no se ha podido implementar de manera óptima. Para salvar estos inconvenientes, se han llevado a cabo medidas pero no aportaban la seguridad que requería una prueba automática a causa de retardos incontrolados y otros factores difíciles de controlar.

Un estudio muy exhaustivo entorno a este dispositivo y a esta medida en particular podría salvar los inconvenientes y/o dar soluciones eficientes para poder llevar a cabo la automatización de la medida.

Para finalizar, un avance muy interesante sería montar un banco de pruebas basado en sistemas tipo Raspberry Pi o similares. Con esto, si se dispone de instrumentos de medidas similares a los usados en este proyecto, se podría transportar el banco de pruebas realizando pocos cambios en el código.

Apéndices

Apéndices A

Códigos fuente más importantes del DVT

A.1. Introducción

En el presente apéndice se pretende mostrar los principales códigos fuentes del DVT.

A.2. Código fuente del DVT (DVT.txt)

```
*** Settings ***
Force Tags        DVT
Library           Dialogs
Library           libs/libraries/testlibraries/
   TransferFunction.py
Library           libs/libraries/testlibraries/
   HtmlReportLibrary.py
Resource          Multi_Tone_Power_Ratio.txt
Resource          Noise_Power_Spectral_Density.txt
Resource          libs/resources/
   ArbitraryWaveformGeneratorVisaResource.txt
```

```
Resource          libs/resources/PowerSupplyVisaResource.
    txt
Resource          libs/resources/
    TemperatureForceSystemVisaResource.txt
Resource          libs/resources/DataLoggerVisaResource.
    txt
Resource          libs/resources/
    DigitalOscilloscopeVisaResource.txt
Resource          libs/resources/
    SpectrumAnalyzerVisaResource.txt
Resource          libs/resources/SPIResource.txt

*** Test Cases ***
Murviter-DVT
    [Tags]    Murviter
    [Setup]    Setup - Teardown
    ${channels}=    Set variable    ${CHANNELS_TO_RUN}
    ${time_aux}=    Get Time    year month day hour min
        secs
    ${time}=    Set Variable    ${time_aux[0]}${time_aux
        [1]}${time_aux[2]}_${time_aux[3]}${time_aux[4]}${
        time_aux[5]}
    ${report_filename}=    Set Variable    ${OUTPUTDIR}/${
        DUT}_${time}.html
    Write HTML Title Helper    ${report_filename}
    ##############
    Run Keyword If    ${channels[0]} == 1    Channel    ${
        DUT}    ${time}    TXA    ${report_filename}
    ...    ${ENABLE_TEMP}
    Run Keyword If    ${channels[1]} == 1    Channel    ${
        DUT}    ${time}    TXB    ${report_filename}
    ...    ${ENABLE_TEMP}
```

```
    Run Keyword If    ${channels[2]} == 1    Channel    ${
       DUT}    ${time}    RXA    ${report_filename}
    ...    ${ENABLE_TEMP}
    Run Keyword If    ${channels[3]} == 1    Channel    ${
       DUT}    ${time}    RXB    ${report_filename}
    ...    ${ENABLE_TEMP}
    [Teardown]    Setup - Teardown

*** Keywords ***
Setup - Teardown
    ${id}    ${name}=    Open session SA    ${
       SPECTRUM_ANALYZERS[1]}
    Close session SA    ${id}
    ${id}    ${name}=    Open session AWG    ${
       ARBITRARY_WAVEFORM_GENERATORS[0]}
    Close session AWG    ${id}
    ${id}    ${name}=    Open session PS    ${
       POWER_SUPPLIES[0]}
    Close session PS    ${id}
    ${id}    ${name}=    Run Keyword If    ${ENABLE_TEMP}
       == 1    Open session TFS    ${TEMPERATURE_FS[0]}
    Run Keyword If    ${ENABLE_TEMP} == 1    Close session
        TFS    ${id}

Channel
    [Arguments]    ${dut}    ${time}    ${channel}    ${
       report_filename}    ${enable_temp}
    ${return}=    Run Keyword If    ${ENABLE_TEMP} == 1
          Temperature iterations    ${dut}    ${time}    ${
       channel}
    ...    ELSE    Voltage iterations    ${dut}    ${time}
          ${channel}    NO_TEMP
```

```
Write HTML Header Helper    ${report_filename}    ${
    dut}    ${time}    ${channel}
${list_of_TestVdcIdc}=    Evaluate    [x for x in ${
    return} if x[0]=='TestVdcIdc']
${length_TestVdcIdc}=    Get Length    ${
    list_of_TestVdcIdc}
${pin_namesV}=    Set Variable    ${PIN_NAMESV}
${pin_namesI}=    Set Variable    ${PIN_NAMESI}
Run Keyword If    ${length_TestVdcIdc} > 0    Write
    STATIC MEASUREMENTS HTML Helper    ${report_filename
    }    ${list_of_TestVdcIdc}    ${pin_namesV}    ${
    pin_namesI}
${list_of_TestNoisePSD}=    Evaluate    [x for x in ${
    return} if x[0]=='TestNoisePSD']
${length_TestNoisePSD}=    Get Length    ${
    list_of_TestNoisePSD}
Run keyword if    ${length_TestNoisePSD} > 0    Write
    NOISE PSD IMG HTML Helper    ${report_filename}    $
    {list_of_TestNoisePSD}
${list_of_TestTF}=    Evaluate    [x for x in ${return
    } if x[0]=='TestTF']
${length_TestTF}=    Get Length    ${list_of_TestTF}
Run keyword if    ${length_TestTF} > 0    Write
    TRANSFER FUNCTION IMG HTML Helper    ${
    report_filename}    ${list_of_TestTF}
${list_of_TestDynamicConsumption}=    Evaluate    [x
    for x in ${return} if x[0]=='TestDynamicConsumption
    ']
${length_TestDynamicConsumption}=    Get Length    ${
    list_of_TestDynamicConsumption}
```

```
    Run keyword if    ${length_TestDynamicConsumption} > 0
        Write DYNAMIC CONSUMPTION HTML Helper    ${
       report_filename}    ${list_of_TestDynamicConsumption
       }
    ${list_of_TestMTPR}=    Evaluate    [x for x in ${
       return} if x[0]=='TestMTPR']
    ${length_TestMTPR}=    Get Length    ${
       list_of_TestMTPR}
    Run keyword if    ${length_TestMTPR} > 0    Write MTPR
        IMG HTML Helper    ${report_filename}    ${
       list_of_TestMTPR}
    OperatingSystem.Run    mutt -s "change the channel" ${
       EMAIL_NOTIFICATION} < /dev/null
    Pause Execution    Change the channel

Temperature iterations
    [Arguments]    ${dut}    ${time}    ${channel}
    @{T_range}=    Set Variable    ${T_RANGE}
    ${return}=    Create List
    ####################
    ${id}    ${name}=    Open session TFS    ${
       TEMPERATURE_FS[0]}
    : FOR    ${temp}    IN    @{T_range}
    \    Set temperature    ${id}    1    ${temp}
    \    Set temperature sensor    ${id}    1
    \    Set soaktime    ${id}    1    ${SOAK_TIME}
    \    Start process    ${id}    1
    \    ${valid_temp}=    Check temperature    ${id}    $
       {temp}    2    ${SOAK_TIME}
    \    ${tmp}=    Run Keyword If    ${valid_temp} ==
       True    Voltage iterations    ${dut}    ${time}
    \    ...    ${channel}    ${temp}C
```

```
    \    ${return}=    Combine Lists    ${return}    ${tmp
       }
    Close session TFS    ${id}
    ####################
    [Return]    ${return}

Voltage iterations
    [Arguments]    ${dut}    ${time}    ${channel}    ${
       temp}
    @{V_range}=    Set Variable    ${V_RANGE}
    ${return}=    Create List
    : FOR    ${volt5}    ${volt33}    IN    @{V_range}
    \    ${id}    ${name}=    Open session PS    ${
       POWER_SUPPLIES[0]}
    \    Set voltage PS    ${id}    1    ${volt5}
    \    Sleep    1
    \    Set voltage PS    ${id}    2    ${volt33}
    \    Sleep    1
    \    Set output ON PS    ${id}
    \    ${tmp}=    Run Keyword If    '${channel}' == 'TXA
       '    SPI iterations    ${dut}    ${time}
    \    ...    ${channel}    ${temp}    ${volt5}V    ${
       volt33}V    ${FRAMES_TXA}
    \    ${return}=    Run Keyword if    '${channel}' == '
       TXA'    Combine Lists    ${return}    ${tmp}
    \    ...    ELSE    Combine Lists    ${return}
    \    ${tmp}=    Run Keyword if    '${channel}' == 'TXB
       '    SPI iterations    ${dut}    ${time}
    \    ...    ${channel}    ${temp}    ${volt5}V    ${
       volt33}V    ${FRAMES_TXB}
    \    ${return}=    Run Keyword if    '${channel}' == '
       TXB'    Combine Lists    ${return}    ${tmp}
    \    ...    ELSE    Combine Lists    ${return}
```

```
    \    ${tmp}=    Run Keyword If    '${channel}' == 'RXA
       '    SPI iterations    ${dut}    ${time}
    \    ...    ${channel}    ${temp}    ${volt5}V    ${
       volt33}V    ${FRAMES_RXA}
    \    ${return}=    Run Keyword if    '${channel}' == '
       RXA'    Combine Lists    ${return}    ${tmp}
    \    ...    ELSE    Combine Lists    ${return}
    \    ${tmp}=    Run Keyword If    '${channel}' == 'RXB
       '    SPI iterations    ${dut}    ${time}
    \    ...    ${channel}    ${temp}    ${volt5}V    ${
       volt33}V    ${FRAMES_RXB}
    \    ${return}=    Run Keyword if    '${channel}' == '
       RXB'    Combine Lists    ${return}    ${tmp}
    \    ...    ELSE    Combine Lists    ${return}
    Close session PS    ${id}
    [Return]    ${return}

SPI iterations
    [Arguments]    ${dut}    ${time}    ${channel}    ${
       temp}    ${volt5}    ${volt33}
    ...    ${frames}
    @{frames}=    Set variable    ${frames}
    ${return}=    Create List
    : FOR    ${frame}    IN    @{frames}
    \    ${tmp}=    Tests    ${dut}    ${time}    ${
       channel}    ${temp}
    \    ...    ${volt5}    ${volt33}    ${frame}
    \    ${return}=    Combine Lists    ${return}    ${tmp
       }
    [Return]    ${return}

Tests
```

```
[Arguments]    ${dut}    ${time}    ${channel}    ${
   temp}    ${volt5}    ${volt33}
...    ${frame}
@{frame}=    Set variable    ${frame}
${return}=    Create List
${ChipID}=    TX and RX SPI frames    ${MODEMS[0]["pc
   "]}    00000000    00000000    00000000    00000000
: FOR    ${Gain}    ${SPI_11}    ${SPI_01}    ${SPI_10
   }    ${SPI_00}    ${output_volt_AWG_TF}
...    ${att_SA_TF}    ${ref_level_SA_TF}    ${
   output_volt_AWG_MTPR}    ${att_SA_MTPR}    ${
   ref_level_SA_MTPR}    ${DC_test}
...    ${Noise_PSD_test}    ${TF_test}    ${
   Dynamic_Consumption_test}    ${MTPR_test}    ${
   STB_test}    IN
...    @{frame}
\    ${SPIout}=    TX and RX SPI frames    ${MODEMS
   [0]["pc"]}    ${SPI_11}    ${SPI_01}    ${SPI_10}
\    ...    ${SPI_00}
\    ${tmp}=    Run Keyword If    ${DC_test}== 1
   TEMPLATE - Test Vdc and Idc    ${dut}    ${time}
\    ...    ${channel}    ${temp}    ${volt5}    ${
   volt33}    ${Gain}
\    ...    ${SPI_11}    ${SPI_01}    ${SPI_10}    ${
   SPI_00}
\    Run Keyword If    ${DC_test}== 1    Append To
   List    ${return}    ${tmp}
\    ${tmp}=    Run Keyword If    ${Noise_PSD_test}==
   1    TEMPLATE - Test Noise PSD    ${dut}    ${time}
\    ...    ${channel}    ${temp}    ${volt5}    ${
   volt33}    ${Gain}
\    ...    ${SPI_11}    ${SPI_01}    ${SPI_10}    ${
   SPI_00}
```

```
\    Run Keyword If    ${Noise_PSD_test}== 1    Append
    To List    ${return}    ${tmp}
\    ${tmp}=    Run Keyword If    ${TF_test}== 1
   TEMPLATE - Test Transfer Function    ${dut}    ${
   time}
\    ...    ${channel}    ${temp}    ${volt5}    ${
   volt33}    ${Gain}
\    ...    ${SPI_11}    ${SPI_01}    ${SPI_10}    ${
   SPI_00}    ${output_volt_AWG_TF}
\    ...    ${att_SA_TF}    ${ref_level_SA_TF}
\    Run Keyword If    ${TF_test}== 1    Append To
   List    ${return}    ${tmp}
\    ${tmp}=    Run Keyword If    ${
   Dynamic_Consumption_test}== 1    TEMPLATE - Test
   Dynamic Consumption    ${dut}    ${time}
\    ...    ${channel}    ${temp}    ${volt5}    ${
   volt33}    ${Gain}
\    ...    ${SPI_11}    ${SPI_01}    ${SPI_10}    ${
   SPI_00}    ${output_volt_AWG_MTPR}
\    Run Keyword If    ${Dynamic_Consumption_test}== 1
      Append To List    ${return}    ${tmp}
\    ${tmp}=    Run Keyword If    ${MTPR_test}== 1
   TEMPLATE - Test MTPR    ${dut}    ${time}
\    ...    ${channel}    ${temp}    ${volt5}    ${
   volt33}    ${Gain}
\    ...    ${SPI_11}    ${SPI_01}    ${SPI_10}    ${
   SPI_00}    ${output_volt_AWG_MTPR}
\    ...    ${att_SA_MTPR}    ${ref_level_SA_MTPR}
\    Run Keyword If    ${MTPR_test}== 1    Append To
   List    ${return}    ${tmp}
\    Comment    Run Keyword If    ${STB_test}== '1'
     TEMPLATE - Test Stability    ${filename}    ${
   STB_OUTPUT_VOLTAGE_AWG}
```

```
    [Return]    ${return}

TEMPLATE - Test Vdc and Idc
    [Arguments]    ${dut}    ${time}    ${channel}    ${
       temp}    ${volt5}    ${volt33}
    ...    ${Gain}    ${SPI_11}    ${SPI_01}    ${SPI_10}
          ${SPI_00}
    ${id}    ${name}=    Open session DL    ${DATA_LOGGERS
       [1]}
    ${measures_Vdc}=    Get Vdc @ CH101:CH120    ${id}
    ${measures_Idc}=    Get Idc @ CH121:CH122    ${id}
    Close session DL    ${id}
    # Generate data file
    ${filename}=    Set Variable    ${OUTPUTDIR}/${dut}_${
       time}_${channel}_${temp}_${volt5}_${volt33}_${SPI_11
       }_${SPI_01}_${SPI_10}_${SPI_00}
    : FOR    ${v}    IN RANGE    0    20
    \    OperatingSystem.Append To File    ${filename}
       _Vdc_Idc.meas    Vdc${v+1}\t\t${measures_Vdc[${v}]}\
       n
    OperatingSystem.Append To File    ${filename}_Vdc_Idc.
       meas    Idc21\t\t${measures_Idc[0]}\n
    OperatingSystem.Append To File    ${filename}_Vdc_Idc.
       meas    Idc22\t\t${measures_Idc[1]}\n
    # Generate setup file
    OperatingSystem.Append To File    ${filename}_Vdc_Idc.
       setup    DATA LOGGER: "${name}"\n
    OperatingSystem.Append To File    ${filename}_Vdc_Idc.
       setup    CONF_1: VOLTAGE=INPUT:IMPEDANCE:AUTO ON,(
       @101:120);MEAS:VOLT:DC? (@101:120)\n
    OperatingSystem.Append To File    ${filename}_Vdc_Idc.
       setup    CONF_2: CURRENT=MEAS:CURR:DC? (@121:122)\n
```

```
    OperatingSystem.Append To File    ${filename}_Vdc_Idc.
       setup    ${SPI_11}_${SPI_01}_${SPI_10}_${SPI_00}\n
    # Return
    ${return}=    Create List    TestVdcIdc    ${temp}
       ${volt5}    ${volt33}    ${Gain}
    ...    ${SPI_11}    ${SPI_10}    ${SPI_01}    ${SPI_00
       }
    : FOR    ${v}    IN RANGE    0    20    #0..19
    \    Append To List    ${return}    ${measures_Vdc[${v
       }]}
    : FOR    ${i}    IN RANGE    0    2    #0,1
    \    Append To List    ${return}    ${measures_Idc[${i
       }]}
    [Return]    ${return}

TEMPLATE - Test Noise PSD
    [Arguments]    ${dut}    ${time}    ${channel}    ${
       temp}    ${volt5}    ${volt33}
    ...    ${Gain}    ${SPI_11}    ${SPI_01}    ${SPI_10}
          ${SPI_00}
    # Open session
    ${id}    ${name}=    Open session SA    ${
       SPECTRUM_ANALYZERS[1]}
    # Set configuration in spectrum analyzer
    Set configuration for Noise PSD    ${id}    ${
       NOISE_ATTENUATION}    ${NOISE_REFERENCE_LEVEL}    ${
       NOISE_START_FREQUENCY}    ${NOISE_STOP_FREQUENCY}
          ${NOISE_RESOLUTION_BANDWIDTH}
    ...    ${NOISE_VIDEO_BANDWIDTH}    ${
       NOISE_REFERENCE_LEVEL_OFFSET}
    # Capture trace
    ${detector}=    Set Variable    ${NOISE_DETECTOR}
```

```
${frequencies}    ${data_points}=    Capture Noise PSD
    trace    ${id}    ${detector}
# Close session
Close session SA    ${id}
    # Plot image
${filename}=    Set Variable    ${OUTPUTDIR}/${dut}_${
   time}_${channel}_${temp}_${volt5}_${volt33}_${SPI_11
   }_${SPI_01}_${SPI_10}_${SPI_00}
Plot Noise PSD Trace    ${frequencies}    ${
   data_points}    ${filename}_NoisePSD.png
Log Image    ${filename}_NoisePSD.png
# Generate data file
${length}=    Get Length    ${frequencies}
: FOR    ${i}    IN RANGE    ${length}
\    OperatingSystem.Append To File    ${filename}
   _NoisePSD.meas    ${frequencies[${i}]}\t\t${
   data_points[${i}]}\n
# Generate setup file
OperatingSystem.Append To File    ${filename}_NoisePSD
   .setup    NOISE PSD measurement\n
OperatingSystem.Append To File    ${filename}_NoisePSD
   .setup    SPECTRUM ANALYZER CONFIGURATION\n
OperatingSystem.Append To File    ${filename}_NoisePSD
   .setup    "${name}"\n
OperatingSystem.Append To File    ${filename}_NoisePSD
   .setup    ATT = ${NOISE_ATTENUATION}\n
OperatingSystem.Append To File    ${filename}_NoisePSD
   .setup    REF_LEVEL = ${NOISE_REFERENCE_LEVEL}\n
OperatingSystem.Append To File    ${filename}_NoisePSD
   .setup    START_FREQUENCY = ${NOISE_START_FREQUENCY
   }\n
OperatingSystem.Append To File    ${filename}_NoisePSD
   .setup    STOP_FREQUENCY = ${NOISE_STOP_FREQUENCY}\n
```

```
    OperatingSystem.Append To File    ${filename}_NoisePSD
       .setup    RESOL_BW = ${NOISE_RESOLUTION_BANDWIDTH}\n
    OperatingSystem.Append To File    ${filename}_NoisePSD
       .setup    VIDEO_BW = ${NOISE_VIDEO_BANDWIDTH}\n
    OperatingSystem.Append To File    ${filename}_NoisePSD
       .setup    REF_LEVEL_OFFSET = ${
      NOISE_REFERENCE_LEVEL_OFFSET}\n
    OperatingSystem.Append To File    ${filename}_NoisePSD
       .setup    DETECTOR = ${NOISE_DETECTOR}\n
    OperatingSystem.Append To File    ${filename}_NoisePSD
       .setup    ${SPI_11}_${SPI_01}_${SPI_10}_${SPI_00}\n
    # Return
    ${return}=    Create List    TestNoisePSD    ${temp}
          ${volt5}    ${volt33}    ${Gain}
    ...    ${SPI_11}    ${SPI_10}    ${SPI_01}    ${SPI_00
       }    ${name}    ${NOISE_ATTENUATION}
    ...    ${NOISE_REFERENCE_LEVEL}    ${
       NOISE_START_FREQUENCY}    ${NOISE_STOP_FREQUENCY}
          ${NOISE_RESOLUTION_BANDWIDTH}    ${
       NOISE_VIDEO_BANDWIDTH}    ${
       NOISE_REFERENCE_LEVEL_OFFSET}
    ...    ${detector}    ${filename}_NoisePSD.png
    [Return]    ${return}

TEMPLATE - Test Transfer Function
    [Arguments]    ${dut}    ${time}    ${channel}    ${
       temp}    ${volt5}    ${volt33}
    ...    ${Gain}    ${SPI_11}    ${SPI_01}    ${SPI_10}
          ${SPI_00}    ${output_voltage_AWG}
    ...    ${att_SA}    ${ref_level_SA}
    # Configure AWG with signal to Transfer Function
       mesure
```

```
${id}    ${name}=    Open session AWG    ${
    ARBITRARY_WAVEFORM_GENERATORS[0]}
Change working directory and load file    ${id}    ${
    TF_PATH_AND_FILE_CHIRP}
Set configuration AWG    ${id}    1    ${
    output_voltage_AWG}    ${TF_SAMPLING_RATE_AWG}
Set configuration AWG    ${id}    2    ${
    output_voltage_AWG}    ${TF_SAMPLING_RATE_AWG}
RUN/STOP AWG    ${id}    RUN
Sleep    1
# Configure SA and obtain trace
${id2}    ${name2}=    Open session SA    ${SPECTRUM
    ANALYZERS[1]}
Set configuration for Noise PSD    ${id2}    ${att_SA}
        ${ref_level_SA}    ${TF_START_FREQUENCY}    ${
    TF_STOP_FREQUENCY}    ${TF_RESOLUTION_BANDWIDTH}
...    ${TF_VIDEO_BANDWIDTH}    ${
    TF_REFERENCE_LEVEL_OFFSET}
Sleep    1
# Enable OUTPUT AWG
Enable/Disable Output AWG    ${id}    1    ON
Enable/Disable Output AWG    ${id}    2    ON
Sleep    1
###########
${filename}=    Set Variable    ${OUTPUTDIR}/${dut}_${
    time}_${channel}_${temp}_${volt5}_${volt33}_${SPI_11
    }_${SPI_01}_${SPI_10}_${SPI_00}
${detector}=    Set Variable    ${TF_DETECTOR}
${frequencies}    ${data_points}=    Capture trace
    through PEAK capture    ${id2}    ${detector}
${TF_points}=    Calculate TF    ${data_points}    ${
    channel}    ${Gain}
```

```
Plot TF Trace    ${frequencies}    ${TF_points}    ${
   filename}_TF.png
Log Image    ${filename}_TF.png
Close session AWG    ${id}
# Generate data file
${length}=    Get Length    ${frequencies}
: FOR    ${i}    IN RANGE    ${length}
\    OperatingSystem.Append To File    ${filename}_TF.
   meas    ${frequencies[${i}]}\t\t${TF_points[${i}]}\n
# Generate setup file
OperatingSystem.Append To File    ${filename}_TF.setup
       TRANSFER FUNCTION\n
OperatingSystem.Append To File    ${filename}_TF.setup
       ARBITRARY WAVEFORM GENERATOR\n
OperatingSystem.Append To File    ${filename}_TF.setup
       "${name}"\n
OperatingSystem.Append To File    ${filename}_TF.setup
       SIGNAL = ${TF_PATH_AND_FILE_CHIRP}\n
OperatingSystem.Append To File    ${filename}_TF.setup
       OUTPUT VOLTAGE = ${output_voltage_AWG}V\n
OperatingSystem.Append To File    ${filename}_TF.setup
       SAMPLING RATE = ${TF_SAMPLING_RATE_AWG}Hz\n
OperatingSystem.Append To File    ${filename}_TF.setup
       SPECTRUM ANALYZER CONFIGURATION\n
OperatingSystem.Append To File    ${filename}_TF.setup
       "${name2}"\n
OperatingSystem.Append To File    ${filename}_TF.setup
       ATT = ${att_SA}\n
OperatingSystem.Append To File    ${filename}_TF.setup
       REF_LEVEL = ${ref_level_SA}\n
OperatingSystem.Append To File    ${filename}_TF.setup
       START_FREQUENCY = ${TF_START_FREQUENCY}\n
```

```
    OperatingSystem.Append To File    ${filename}_TF.setup
        STOP_FREQUENCY = ${TF_STOP_FREQUENCY}\n
    OperatingSystem.Append To File    ${filename}_TF.setup
        RESOL_BW = ${TF_RESOLUTION_BANDWIDTH}\n
    OperatingSystem.Append To File    ${filename}_TF.setup
        VIDEO_BW = ${TF_VIDEO_BANDWIDTH}\n
    OperatingSystem.Append To File    ${filename}_TF.setup
        REF_LEVEL_OFFSET = ${TF_REFERENCE_LEVEL_OFFSET}\
      n
    OperatingSystem.Append To File    ${filename}_TF.setup
        DETECTOR = ${TF_DETECTOR}\n
    OperatingSystem.Append To File    ${filename}_TF.setup
        ${SPI_11}_${SPI_01}_${SPI_10}_${SPI_00}\n
    # Return
    ${return}=    Create List    TestTF    ${temp}    ${
      volt5}    ${volt33}    ${Gain}
    ...    ${SPI_11}    ${SPI_10}    ${SPI_01}    ${SPI_00
      }    ${name2}    ${att_SA}
    ...    ${ref_level_SA}    ${TF_START_FREQUENCY}    ${
      TF_STOP_FREQUENCY}    ${TF_RESOLUTION_BANDWIDTH}
      ${TF_VIDEO_BANDWIDTH}    ${TF_REFERENCE_LEVEL_OFFSET
      }
    ...    ${detector}    ${name}    ${
      TF_PATH_AND_FILE_CHIRP}    ${output_voltage_AWG}
      ${TF_SAMPLING_RATE_AWG}    ${filename}_TF.png
    [Return]    ${return}

TEMPLATE - Test Dynamic Consumption
    [Arguments]    ${dut}    ${time}    ${channel}    ${
      temp}    ${volt5}    ${volt33}
    ...    ${Gain}    ${SPI_11}    ${SPI_01}    ${SPI_10}
        ${SPI_00}    ${output_voltage_AWG}
    # Configure AWG with signal to MTPR
```

```
${id}    ${name}=    Open session AWG    ${
    ARBITRARY_WAVEFORM_GENERATORS[0]}
Change working directory and load file    ${id}    ${
    MTPR_PATH_AND_FILE_OFDM}
Set configuration AWG    ${id}    1    ${
    output_voltage_AWG}    ${MTPR_SAMPLING_RATE_AWG}
Set configuration AWG    ${id}    2    ${
    output_voltage_AWG}    ${MTPR_SAMPLING_RATE_AWG}
RUN/STOP AWG    ${id}    RUN
Enable/Disable Output AWG    ${id}    1    ON
Enable/Disable Output AWG    ${id}    2    ON
Sleep    1
# Data Logger
${id2}    ${name2}=    Open session DL    ${
    DATA_LOGGERS[1]}
${measures_Idc}=    Get Idc @ CH121:CH122    ${id2}
Close session DL    ${id2}
Close session AWG    ${id}
# Generate data file
${filename}=    Set Variable    ${OUTPUTDIR}/${dut}_${
    time}_${channel}_${temp}_${volt5}_${volt33}_${SPI_11
    }_${SPI_01}_${SPI_10}_${SPI_00}
OperatingSystem.Append To File    ${filename}
    _Dynamic_Consumption.meas    Idc33\t\t${measures_Idc
    [0]}\n
OperatingSystem.Append To File    ${filename}
    _Dynamic_Consumption.meas    Idc5\t\t${measures_Idc
    [1]}\n
# Generate setup file
OperatingSystem.Append To File    ${filename}
    _Dynamic_Consumption.setup    DATA LOGGER: "${name
    }"\n
```

```
    OperatingSystem.Append To File    ${filename}
       _Dynamic_Consumption.setup    CONF_2: CURRENT=MEAS:
       CURR:DC? (@121:122)\n
    OperatingSystem.Append To File    ${filename}
       _Dynamic_Consumption.setup    ${SPI_11}_${SPI_01}_${
       SPI_10}_${SPI_00}\n
    # Return
    ${return}=    Create List    TestDynamicConsumption
          ${temp}    ${volt5}    ${volt33}    ${Gain}
    ...    ${SPI_11}    ${SPI_10}    ${SPI_01}    ${SPI_00
       }
    : FOR    ${i}    IN RANGE    0    2    #0,1
    \    Append To List    ${return}    ${measures_Idc[${i
       }]}
    [Return]    ${return}

TEMPLATE - Test MTPR
    [Arguments]    ${dut}    ${time}    ${channel}    ${
       temp}    ${volt5}    ${volt33}
    ...    ${Gain}    ${SPI_11}    ${SPI_01}    ${SPI_10}
          ${SPI_00}    ${output_voltage_AWG}
    ...    ${att_SA}    ${ref_level_SA}
    # Configure AWG with signal to MTPR
    ${id}    ${name}=    Open session AWG    ${
       ARBITRARY_WAVEFORM_GENERATORS[0]}
    Change working directory and load file    ${id}    ${
       MTPR_PATH_AND_FILE_OFDM}
    Set configuration AWG    ${id}    1    ${
       output_voltage_AWG}    ${MTPR_SAMPLING_RATE_AWG}
    Set configuration AWG    ${id}    2    ${
       output_voltage_AWG}    ${MTPR_SAMPLING_RATE_AWG}
    RUN/STOP AWG    ${id}    RUN
    # Generate setup file
```

```
${filename}=    Set Variable    ${OUTPUTDIR}/${dut}_${
    time}_${channel}_${temp}_${volt5}_${volt33}_${SPI_11
    }_${SPI_01}_${SPI_10}_${SPI_00}
OperatingSystem.Append To File    ${filename}_MTPR.
    setup    MTPR\n
OperatingSystem.Append To File    ${filename}_MTPR.
    setup    ARBITRARY WAVEFORM GENERATOR\n
OperatingSystem.Append To File    ${filename}_MTPR.
    setup    "${name}"\n
OperatingSystem.Append To File    ${filename}_MTPR.
    setup    SIGNAL = ${MTPR_PATH_AND_FILE_OFDM}\n
OperatingSystem.Append To File    ${filename}_MTPR.
    setup    OUTPUT VOLTAGE = ${output_voltage_AWG}V\n
OperatingSystem.Append To File    ${filename}_MTPR.
    setup    SAMPLING RATE = ${MTPR_SAMPLING_RATE_AWG}Hz
    \n
# Start MTPR mesure
${name2}    ${detector}=    MTPR mesure    ${id}    ${
    filename}    ${ref_level_SA}    ${att_SA}
...    # Open session, obtain MTPR mesure and close
    session with SP
# Close session AWG
Close session AWG    ${id}
# Return
${return}=    Create List    TestMTPR    ${temp}    ${
    volt5}    ${volt33}    ${Gain}
...    ${SPI_11}    ${SPI_10}    ${SPI_01}    ${SPI_00
    }    ${name2}    ${att_SA}
...    ${ref_level_SA}    ${MTPR_START_FREQUENCY}    $
    {MTPR_STOP_FREQUENCY}    ${MTPR_RESOLUTION_BANDWIDTH
    }    ${MTPR_VIDEO_BANDWIDTH}    ${
    MTPR_REFERENCE_LEVEL_OFFSET}
```

```
    ...    ${detector}    ${name}    ${
       MTPR_PATH_AND_FILE_OFDM}    ${output_voltage_AWG}
          ${MTPR_SAMPLING_RATE_AWG}    ${filename}_MTPR.png
    [Return]    ${return}

TEMPLATE - Test Stability
    [Arguments]    ${dut}    ${time}    ${channel}    ${
       temp}    ${volt5}    ${volt33}
    ...    ${Gain}    ${SPI_11}    ${SPI_01}    ${SPI_10}
          ${SPI_00}    ${output_voltage_AWG}
    # Configure AWG with signal to Transfer Function
       mesure
    ${id}    ${name}=    Open session AWG    ${
       ARBITRARY_WAVEFORM_GENERATORS[0]}
    Change working directory and load file    ${id}    ${
       STB_PATH_AND_FILE_SQUARE}
    Set configuration AWG    ${id}    1    ${
       output_voltage_AWG}    ${STB_SAMPLING_RATE_AWG}
    Set configuration AWG    ${id}    2    ${
       output_voltage_AWG}    ${STB_SAMPLING_RATE_AWG}
    RUN/STOP AWG    ${id}    RUN
    Enable/Disable Output AWG    ${id}    1    ON
    Enable/Disable Output AWG    ${id}    2    ON
    # Obtain trace
    ${id2}    ${name2}=    Open session DO    ${
       DIGITAL_OSCILLOSCOPES[0]}
    Set acquisition mode average    ${id2}    ${
       STB_NUM_AVERAGES}
    Set Vertical and Horizontal Scale    ${id2}    1    ${
       STB_VERTICAL_SCALE_1}    ${STB_HORIZONTAL_SCALE}
    Set Vertical and Horizontal Scale    ${id2}    2    ${
       STB_VERTICAL_SCALE_2}    ${STB_HORIZONTAL_SCALE}
    Set channel offset    ${id2}    1    1.15
```

```
Set channel offset    ${id2}    2    1.18
Get fall time    ${id2}
Get rise time    ${id2}
Get positive overshoot    ${id2}
Get negative overshoot    ${id2}
Sleep    2
Save snapshot    ${id2}    ${filename}_STB.png
Close session DO    ${id2}
# Close session AWG
Close session AWG    ${id}
# Generate setup file
${filename}=    Set Variable    ${OUTPUTDIR}/${dut}_${
   time}_${channel}_${temp}_${volt}_${SPI_11}_${SPI_01}
   _${SPI_10}_${SPI_00}
OperatingSystem.Append To File    ${filename}_STB.
   setup    STABILITY\n
OperatingSystem.Append To File    ${filename}_STB.
   setup    ARBITRARY WAVEFORM GENERATOR\n
OperatingSystem.Append To File    ${filename}_STB.
   setup    "${name}"\n
OperatingSystem.Append To File    ${filename}_STB.
   setup    SIGNAL = ${STB_PATH_AND_FILE_SQUARE}\n
OperatingSystem.Append To File    ${filename}_STB.
   setup    OUTPUT VOLTAGE = ${STB_OUTPUT_VOLTAGE_AWG}V
   \n
OperatingSystem.Append To File    ${filename}_STB.
   setup    SAMPLING RATE = ${STB_SAMPLING_RATE_AWG}Hz\
   n
OperatingSystem.Append To File    ${filename}_STB.
   setup    DIGITAL OSCILLOSCOPE\n
OperatingSystem.Append To File    ${filename}_STB.
   setup    "${name2}"\n
```

```
OperatingSystem.Append To File    ${filename}_STB.
    setup    NUM_AVG = ${STB_NUM_AVERAGES}\n
OperatingSystem.Append To File    ${filename}_STB.
    setup    VERTICAL_SCALE_1 = ${STB_VERTICAL_SCALE_1}\
    n
OperatingSystem.Append To File    ${filename}_STB.
    setup    VERTICAL_SCALE_2 = ${STB_VERTICAL_SCALE_2}\
    n
OperatingSystem.Append To File    ${filename}_STB.
    setup    HORIZONTAL_SCALE = ${STB_HORIZONTAL_SCALE}\
    n
OperatingSystem.Append To File    ${filename}_STB.
    setup    ${SPI_11}_${SPI_01}_${SPI_10}_${SPI_00}\n
# Return
${return}=    Create List    TestSTB    ${temp}    ${
    volt5}    ${volt33}    ${Gain}
...    ${SPI_11}    ${SPI_10}    ${SPI_01}    ${SPI_00
    }    ${name2}    ${STB_VERTICAL_SCALE_1}
...    ${STB_VERTICAL_SCALE_2}    ${
    STB_HORIZONTAL_SCALE}    ${STB_NUM_AVERAGES}    ${
    name}    ${STB_PATH_AND_FILE_CHIRP}    ${
    output_voltage_AWG}
...    ${STB_SAMPLING_RATE_AWG}    ${filename}_STB.png
[Teardown]
```

A.3. Código fuente del cálculo de MTPR (MultiTonePowerRatio.py)

Para el cálculo de MTPR en cada notch, el algoritomo realiza lo siguiente:

1. Frecuencia central = frecuencia del notch
2. $Span = 5 \cdot (separaci'onportadoras)$ por tanto se quedan dos portadoras a cada lado

3. Abre una ventana muy estrecha entorno a la primera portadora y obtiene el valor máximo (A)

4. Abre una ventana muy estrecha entorno a la frecuencia central (notch) y obtiene el valor máximo (B)

5. Realiza la operación $A - B$

```
#!/usr/bin/python

import os
import re
from numpy import *
import Gnuplot

class MultiTonePowerRatio:

    def __init__(self):
        pass

    def calculate_MTPR(self, freq, power):
        freq  = [float(x) for x in freq]
        power = [float(x) for x in power]

        top    = max(power[X40:X60])
        bottom = max(power[X40:X60])

        top_index    = power.index(top)
        bottom_index = power.index(bottom)

        return  ([freq[bottom_index],top-bottom],top)
```

A.4. Código fuente librería de generación del informe (HtmlReportLibrary.py)

```
# -*- coding: utf-8 -*-
#!/usr/bin/python

import os
import re
import HTML

class HtmlReportLibrary:

    def __init__(self):
        pass

    def Write_HTML_title_helper(self, path_and_filename):
        f = open(path_and_filename,"a")
        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.write("""<TITLE>DVT REPORT</TITLE>\n<H1 FONT
            FACE ="arial">DVT REPORT</H1>\n""")
        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.close()

    def Write_HTML_header_helper(self, path_and_filename,
        sample,time,channel):
        f = open(path_and_filename,"a")
        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.write("""<P>SAMPLE: """+sample+"""</P>\n""")
        f.write("""<P>TIME: """+time+"""</P>\n""")
        f.write("""<P>CHANNEL: """+channel+"""</P>\n""")
```

```
        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.close()

    def Write_STATIC_MEASUREMENTS_HTML_helper(self,
        path_and_filename, measures, Pin_namesV, Pin_namesI)
        :
        settings_names = ["TEMP","VDD5","VDD33"]
        SPI_names = ["Gain (dB)","SPI_11","SPI_10","SPI_01
            ","SPI_00"]

        f = open(path_and_filename,"a")
        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.write("""<H3>STATIC MEASUREMENTS</H3>\n""")

        for meas in measures:
            settings_value = meas[1:4]
            settings_table = [settings_names,
                settings_value]
            settings_table = [[settings_table[j][i] for j
                in xrange(len(settings_table))] for i in
                xrange(len(settings_table[0]))]
            tablecode = HTML.table(settings_table)
            f.write(tablecode+"\n")
            f.write("""<P></P>\n""")

            SPI_value = meas[4:9]
            aux = ["0x"+x for x in SPI_value[1:]]
            SPI_value = [SPI_value[0]]+aux
            SPI_table = [SPI_names,SPI_value]
```

```
SPI_table = [[SPI_table[j][i] for j in xrange(
    len(SPI_table))] for i in xrange(len(
    SPI_table[0]))]
tablecode = HTML.table(SPI_table)
f.write(tablecode+"\n")
f.write("""<P></P>\n""")

data_valueV = ['%.3f' %float(x) for x in meas
    [9:-2]]
data_valueV = data_valueV[:3]+data_valueV[4:]
data_tableV = [Pin_namesV,data_valueV]
data_tableV = [[data_tableV[j][i] for j in
    xrange(len(data_tableV))] for i in xrange(
    len(data_tableV[0]))]
tablecode = HTML.table(data_tableV,header_row
    =["PIN NAME","VALUE (V)"])
f.write(tablecode+"\n")
f.write("""<P></P>\n""")

data_valueI = [float(x)*1000 for x in meas
    [-2:]]
data_valueI = ['%.3f' %x for x in data_valueI
    ]
data_tableI = [Pin_namesI,data_valueI]
data_tableI = [[data_tableI[j][i] for j in
    xrange(len(data_tableI))] for i in xrange(
    len(data_tableI[0]))]
tablecode = HTML.table(data_tableI,header_row
    =["PIN NAME","VALUE (mA)"])
f.write(tablecode+"\n")
f.write("""<P><HR width="750" align="left"></P
    >\n""")
```

```
        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.close()


    def Write_NOISE_PSD_IMG_HTML_helper(self,
        path_and_filename, measures):
        settings_names = ["TEMP","VDD5","VDD33"]
        SPI_names = ["Gain (dB)","SPI_11","SPI_10","SPI_01
            ","SPI_00"]
        SA_settings_names = ["SP ANALYZER","ATT (dB)","REF
            LEVEL (dBm)","START FREQ (MHz)","STOP FREQ (MHZ
            )","RESOL BW (kHz)","VIDEO BW (kHz)","REF LEVEL
            OFFSET (dB)","DETECTOR"]

        f = open(path_and_filename,"a")
        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.write("""<H3>NOISE PSD</H3>\n""")

        for meas in measures:
            settings_value = meas[1:4]
            settings_table = [settings_names,
                settings_value]
            settings_table = [[settings_table[j][i] for j
                in xrange(len(settings_table))] for i in
                xrange(len(settings_table[0]))]
            tablecode = HTML.table(settings_table)
            f.write(tablecode+"\n")
            f.write("""<P></P>\n""")

            SPI_value = meas[4:9]
            aux = ["0x"+x for x in SPI_value[1:]]
```

```
SPI_value = [SPI_value[0]]+aux
SPI_table = [SPI_names,SPI_value]
SPI_table = [[SPI_table[j][i] for j in xrange(
    len(SPI_table))] for i in xrange(len(
    SPI_table[0]))]
tablecode = HTML.table(SPI_table)
f.write(tablecode+"\n")
f.write("""<P></P>\n""")

SA_settings_value = meas[9:-1]
SA_settings_value[3] = SA_settings_value[3]/1
    e6
SA_settings_value[4] = SA_settings_value[4]/1
    e6
SA_settings_value[5] = SA_settings_value[5]/1
    e3
SA_settings_value[6] = SA_settings_value[6]/1
    e3
SA_settings_table = [SA_settings_names,
    SA_settings_value]
SA_settings_table = [[SA_settings_table[j][i]
    for j in xrange(len(SA_settings_table))] for
     i in xrange(len(SA_settings_table[0]))]
tablecode = HTML.table(SA_settings_table,
    header_row=["SPECTRUM ANALYZER SETTINGS"])
f.write(tablecode+"\n")
f.write("""<P></P>\n""")

f.write("""<IMG SRC="""+str(meas[-1])+""" ALT=
    """+str(meas[-1])+""" width="500" height
    ="500"\n""")
f.write("""<P><HR width="750" align="left"></P
    >\n""")
```

```
        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.close()

    def Write_TRANSFER_FUNCTION_IMG_HTML_helper(self,
        path_and_filename, measures):
        settings_names = ["TEMP","VDD5","VDD33"]
        SPI_names = ["Gain (dB)","SPI_11","SPI_10","SPI_01
            ","SPI_00"]
        SA_settings_names = ["SP ANALYZER","ATT (dB)","REF
            LEVEL (dBm)","START FREQ (MHz)","STOP FREQ (MHZ
            )","RESOL BW (kHz)","VIDEO BW (kHz)","REF LEVEL
            OFFSET (dB)","DETECTOR"]
        AWG_settings_names = ["AWG","SIGNAL","OUTPUT VOLT
            (Vpp)","SAMPLING RATE (GHz)"]

        f = open(path_and_filename,"a")
        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.write("""<H3>TRANSFER FUNCTION</H3>\n""")

        for meas in measures:
            settings_value = meas[1:4]
            settings_table = [settings_names,
                settings_value]
            settings_table = [[settings_table[j][i] for j
                in xrange(len(settings_table))] for i in
                xrange(len(settings_table[0]))]
            tablecode = HTML.table(settings_table)
            f.write(tablecode+"\n")
            f.write("""<P></P>\n""")
```

```
SPI_value = meas[4:9]
aux = ["0x"+x for x in SPI_value[1:]]
SPI_value = [SPI_value[0]]+aux
SPI_table = [SPI_names,SPI_value]
SPI_table = [[SPI_table[j][i] for j in xrange(
    len(SPI_table))] for i in xrange(len(
    SPI_table[0]))]
tablecode = HTML.table(SPI_table)
f.write(tablecode+"\n")
f.write("""<P></P>\n""")

AWG_settings_value = meas[-5:-1]
AWG_settings_value[-2] = '%.3f' %
    AWG_settings_value[-2]
AWG_settings_value[-1] = AWG_settings_value
    [-1]/1e9
AWG_settings_table = [AWG_settings_names,
    AWG_settings_value]
AWG_settings_table = [[AWG_settings_table[j][i
    ] for j in xrange(len(AWG_settings_table))]
    for i in xrange(len(AWG_settings_table[0]))]
tablecode = HTML.table(AWG_settings_table,
    header_row=["AWG_SETTINGS"])
f.write(tablecode+"\n")
f.write("""<P></P>\n""")

SA_settings_value = meas[9:-5]
SA_settings_value[3] = SA_settings_value[3]/1
    e6
SA_settings_value[4] = SA_settings_value[4]/1
    e6
SA_settings_value[5] = SA_settings_value[5]/1
    e3
```

```
            SA_settings_value[6] = SA_settings_value[6]/1
                e3
            SA_settings_table = [SA_settings_names,
                SA_settings_value]
            SA_settings_table = [[SA_settings_table[j][i]
                for j in xrange(len(SA_settings_table))] for
                 i in xrange(len(SA_settings_table[0]))]
            tablecode = HTML.table(SA_settings_table,
                header_row=["SPECTRUM ANALYZER SETTINGS"])
            f.write(tablecode+"\n")
            f.write("""<P></P>\n""")

            f.write("""<IMG SRC="""+str(meas[-1])+""" ALT=
                """+str(meas[-1])+""" width="500" height
                ="500"\n""")
            f.write("""<P><HR width="750" align="left"></P
                >\n""")

        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.close()

    def Write_Dynamic_Consumption_HTML_helper(self,
        path_and_filename, measures):
        settings_names = ["TEMP","VDD5","VDD33"]
        SPI_names = ["Gain (dB)","SPI_11","SPI_10","SPI_01
            ","SPI_00"]
        Pin_namesI = ["I(3V3)","I(5V)"]

        f = open(path_and_filename,"a")
        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.write("""<H3>DYNAMIC CONSUMPTION</H3>\n""")
```

```
for meas in measures:
    settings_value = meas[1:4]
    settings_table = [settings_names,
       settings_value]
    settings_table = [[settings_table[j][i] for j
       in xrange(len(settings_table))] for i in
       xrange(len(settings_table[0]))]
    tablecode = HTML.table(settings_table)
    f.write(tablecode+"\n")
    f.write("""<P></P>\n""")

    SPI_value = meas[4:9]
    aux = ["0x"+x for x in SPI_value[1:]]
    SPI_value = [SPI_value[0]]+aux
    SPI_table = [SPI_names,SPI_value]
    SPI_table = [[SPI_table[j][i] for j in xrange(
       len(SPI_table))] for i in xrange(len(
       SPI_table[0]))]
    tablecode = HTML.table(SPI_table)
    f.write(tablecode+"\n")
    f.write("""<P></P>\n""")

    data_valueI = [float(x)*1000 for x in meas
       [-2:]]
    data_valueI = ['%.3f' %x for x in data_valueI
       ]
    data_tableI = [Pin_namesI,data_valueI]
    data_tableI = [[data_tableI[j][i] for j in
       xrange(len(data_tableI))] for i in xrange(
       len(data_tableI[0]))]
    tablecode = HTML.table(data_tableI,header_row
       =["PIN NAME","VALUE (mA)"])
```

```
            f.write(tablecode+"\n")
            f.write("""<P><HR width="750" align="left"></P
                >\n""")

        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.close()

    def Write_MTPR_IMG_HTML_helper(self, path_and_filename
        , measures):
        settings_names = ["TEMP","VDD5","VDD33"]
        SPI_names = ["Gain (dB)","SPI_11","SPI_10","SPI_01
            ","SPI_00"]
        SA_settings_names = ["SP ANALYZER","ATT (dB)","REF
            LEVEL (dBm)","START FREQ (MHz)","STOP FREQ (MHZ
            )","RESOL BW (kHz)","VIDEO BW (kHz)","REF LEVEL
            OFFSET (dB)","DETECTOR"]
        AWG_settings_names = ["AWG","SIGNAL","OUTPUT VOLT
            (Vpp)","SAMPLING RATE (GHz)"]
        Consumption_names = ["I(3V3)","I(5V)"]

        f = open(path_and_filename,"a")
        f.write("""<HR NOSHADE size=3 width="750" align="
            left">\n""")
        f.write("""<H3>MTPR</H3>\n""")

        for meas in measures:
            settings_value = meas[1:4]
            settings_table = [settings_names,
                settings_value]
            settings_table = [[settings_table[j][i] for j
                in xrange(len(settings_table))] for i in
                xrange(len(settings_table[0]))]
```

```
tablecode = HTML.table(settings_table)
f.write(tablecode+"\n")
f.write("""<P></P>\n""")

SPI_value = meas[4:9]
aux = ["0x"+x for x in SPI_value[1:]]
SPI_value = [SPI_value[0]]+aux
SPI_table = [SPI_names,SPI_value]
SPI_table = [[SPI_table[j][i] for j in xrange(
    len(SPI_table))] for i in xrange(len(
    SPI_table[0]))]
tablecode = HTML.table(SPI_table)
f.write(tablecode+"\n")
f.write("""<P></P>\n""")

AWG_settings_value = meas[18:22]
AWG_settings_value[-2] = '%.3f' %
    AWG_settings_value[-2]
AWG_settings_value[-1] = AWG_settings_value
    [-1]/1e9
AWG_settings_table = [AWG_settings_names,
    AWG_settings_value]
AWG_settings_table = [[AWG_settings_table[j][i
    ] for j in xrange(len(AWG_settings_table))]
    for i in xrange(len(AWG_settings_table[0]))]
tablecode = HTML.table(AWG_settings_table,
    header_row=["AWG_SETTINGS"])
f.write(tablecode+"\n")
f.write("""<P></P>\n""")

SA_settings_value = meas[9:18]
SA_settings_value[3] = SA_settings_value[3]/1
    e6
```

```
        SA_settings_value[4] = SA_settings_value[4]/1
            e6
        SA_settings_value[5] = SA_settings_value[5]/1
            e3
        SA_settings_value[6] = SA_settings_value[6]/1
            e3
        SA_settings_table = [SA_settings_names,
            SA_settings_value]
        SA_settings_table = [[SA_settings_table[j][i]
            for j in xrange(len(SA_settings_table))] for
             i in xrange(len(SA_settings_table[0]))]
        tablecode = HTML.table(SA_settings_table,
            header_row=["SPECTRUM ANALYZER SETTINGS"])
        f.write(tablecode+"\n")
        f.write("""<P></P>\n""")

        f.write("""<IMG SRC="""+str(meas[-1])+""" ALT=
            """+str(meas[-1])+""" width="500" height
            ="500"\n""")
        f.write("""<P><HR width="750" align="left"></P
            >\n""")

f.write("""<HR NOSHADE size=3 width="750" align="
    left">\n""")
f.close()
```

Apéndices B

Ficheros de entrada del DVT y ejecución

B.1. Introducción

El presente apéndice pretende mostrar la estructura de los siguientes ficheros: fichero de configuración del DVT (settings_dvt.py), fichero de declaración del banco de pruebas (analog_bench.py) y fichero de instrumentos (external_equipment.py).

Hay que destacar que todo el sistema de caracterización (incluida la GUI) ha sido desarrollado para trabajar bajo un entorno linux.

B.2. Ejecución del DVT

Una vez repasados los ajustes y definidos los casos de test en el fichero de configuración, se procedería a ejecutar el DVT. Para la ejecución del DVT, abrimos una consola en el PC lanzador y ejecutamos el siguiente comando:

```
pybot --loglevel NONE --variablefile 'dir1'/analog_bench.py
--variablefile 'dir2'/settings_dvt.py --outputdir 'dir3'
-test Tests.Analogic.Murviter-DVT.Murviter-DVT 'dir_test'
```

'Dir*' representa el directorio donde se encuentra el respectivo fichero. Los ficheros de salida (ficheros individuales de cada medida e informe final) se generan en el directorio especificado en outputdir y siguen el siguiente patrón: NombreMuestra_fecha_hora_canal_temp_volt5_volt33_SPIFrame_NombreMedida.xxx.

B.3. Fichero de configuración (settings_dvt.py)

```
#!/usr/bin/python
import math

##############################################################
########### GLOBAL SETTINGS & SPI FRAMES ###############
##############################################################
#NOTE:
#ENABLE  == 1
#DISABLE == 0

#OTHER CODIFICATION IS INVALID!!!

CHANNELS_TO_RUN = [1,1,1,1]
#CHANNELS_TO_RUN[0] = TXA
#CHANNELS_TO_RUN[1] = TXB
#CHANNELS_TO_RUN[2] = RXA
#CHANNELS_TO_RUN[3] = RXB

DUT = "SAMPLE_NAME"
#DUT = "TTT","SSS","FFF",...

EMAIL_NOTIFICATION = "XXX@marvell.com"

ENABLE_TEMP = 1
```

```
#If ENABLE_TEMP = 1 T_RANGE is considered. If ENABLE_TEMP
    = 0 T_RANGE is not considered

T_RANGE = [-XX,X5,XX5]
#Range of temperatures (celsius)

SOAK_TIME = X

V_RANGE = [X,X.X]
#Range of voltages --> [VDD5,VDD33,VDD5,VDD33,...]

PIN_NAMESV = ["NAME1","NAME2","...","NAME20"]
#Data Logger names asociated to data logger channels --> [
    CH1, CH2, CH3...]
#[CH1, CH2, CH3, CH5, CH6...] CH4 is not connected in Data
     Logger.

PIN_NAMESI = ["NAME1","NAME2"]
#Data Logger names asociated to data logger channels --> [
    CH21, CH22]

########## TEST TX ##########
ATT_SA_TX_X_TF = X0
ATT_SA_TX_X_TF = X0

REF_LEVEL_SA_TX_X_TF  = X
REF_LEVEL_SA_TX_X_TF  = X

OUT_VOLT_AWG_TX_TF = X.05

ATT_SA_TX_X_MTPR = X0
ATT_SA_TX_X_MTPR = X0
```

```
REF_LEVEL_SA_TX_X_MTPR = X
REF_LEVEL_SA_TX_X_MTPR = X

OUT_VOLT_AWG_TX_X_MTPR = X
OUT_VOLT_AWG_TX_X_MTPR = X

#Example:
#frame_Gain_TXx     = [Gain,"frame1","frame2","frame3","
    frame4",OUT_VOLT_AWG_TX_TF, ATT_SA_TX_Gain_TF,
    REF_LEVEL_SA_TX_Gain_TF, OUT_VOLT_AWG_TX_Gain_MTPR,
    ATT_SA_TX_Gain_MTPR, REF_LEVEL_SA_TX_Gain_MTPR,
    ENABLE_DC_POINTS, ENABLE_NOISE_PSD,
    ENABLE_TRANSFER_FUNCTION, ENABLE_DYNAMIC_CONSUMPTION,
    ENABLE_MTPR, ENABLE_STABILITY]

##### TX_A #####
frame_X_TXA     = [X,"XXXXXXXX","XXXXXXXX","XXXXXXXX","
    XXXXXXXX",OUT_VOLT_AWG_TX_TF, ATT_SA_TX_X_TF,
    REF_LEVEL_SA_TX_X_TF, OUT_VOLT_AWG_TX_X_MTPR,
    ATT_SA_TX_X_MTPR, REF_LEVEL_SA_TX_X_MTPR, 1, 1, 1, 0, 0, 0]
frame_X_TXA     = [X,"XXXXXXXX","XXXXXXXX","XXXXXXXX","
    XXXXXXXX",OUT_VOLT_AWG_TX_TF, ATT_SA_TX_X_TF,
    REF_LEVEL_SA_TX_X_TF, OUT_VOLT_AWG_TX_X_MTPR,
    ATT_SA_TX_X_MTPR, REF_LEVEL_SA_TX_X_MTPR, 1, 0, 1, 1, 1, 0]

FRAMES_TXA = [frame_X_TXA, frame_X_TXA]

##### TX_B #####
frame_X_TXB     = [X,"XXXXXXXX","XXXXXXXX","XXXXXXXX","
    XXXXXXXX",OUT_VOLT_AWG_TX_TF, ATT_SA_TX_X_TF,
    REF_LEVEL_SA_TX_X_TF, OUT_VOLT_AWG_TX_X_MTPR,
    ATT_SA_TX_X_MTPR, REF_LEVEL_SA_TX_X_MTPR, 1, 1, 1, 0, 0, 0]
```

```
frame_X_TXB     = [X,"XXXXXXXX","XXXXXXXX","XXXXXXXX","
    XXXXXXXX",OUT_VOLT_AWG_TX_TF,ATT_SA_TX_X_TF,
    REF_LEVEL_SA_TX_X_TF,OUT_VOLT_AWG_TX_X_MTPR,
    ATT_SA_TX_X_MTPR,REF_LEVEL_SA_TX_X_MTPR,1,0,1,1,1,0]

FRAMES_TXB = [frame_X_TXB,frame_X_TXB]


########## TEST RX ##########

ATT_SA_RX_TF = X0

REF_LEVEL_SA_RX_TF = X

OUT_VOLT_AWG_RX_X_TF = X
OUT_VOLT_AWG_RX_X_TF = X

ATT_SA_RX_MTPR = X0

REF_LEVEL_SA_RX_MTPR = -X5

OUT_VOLT_AWG_RX_X_MTPR = X.5
OUT_VOLT_AWG_RX_X_MTPR = X.5

##### RX_A #####
frame_X_RXA   = [X,"XXXXXXXX","XXXXXXXX","XXXXXXXX","
    XXXXXXXX",OUT_VOLT_AWG_RX_X_TF,ATT_SA_RX_TF,
    REF_LEVEL_SA_RX_TF,OUT_VOLT_AWG_RX_X_MTPR,ATT_SA_RX_MTPR
    ,REF_LEVEL_SA_RX_MTPR,1,1,1,1,1,0]
frame_X_RXA   = [X,"XXXXXXXX","XXXXXXXX","XXXXXXXX","
    XXXXXXXX",OUT_VOLT_AWG_RX_X_TF,ATT_SA_RX_TF,
    REF_LEVEL_SA_RX_TF,OUT_VOLT_AWG_RX_X_MTPR,ATT_SA_RX_MTPR
    ,REF_LEVEL_SA_RX_MTPR,1,0,1,1,0,0]
```

```
FRAMES_RXA = [frame_X_RXA, frame_X_RXA]

##### RX_B #####
frame_X_RXB  = [X,"XXXXXXXX","XXXXXXXX","XXXXXXXX","
    XXXXXXXX",OUT_VOLT_AWG_RX_X_TF,ATT_SA_RX_TF,
    REF_LEVEL_SA_RX_TF,OUT_VOLT_AWG_RX_X_MTPR,ATT_SA_RX_MTPR
    ,REF_LEVEL_SA_RX_MTPR,1,1,1,1,0]
frame_X_RXB  = [X,"XXXXXXXX","XXXXXXXX","XXXXXXXX","
    XXXXXXXX",OUT_VOLT_AWG_RX_X_TF,ATT_SA_RX_TF,
    REF_LEVEL_SA_RX_TF,OUT_VOLT_AWG_RX_X_MTPR,ATT_SA_RX_MTPR
    ,REF_LEVEL_SA_RX_MTPR,1,0,1,1,0,0]

FRAMES_RXB = [frame_X_RXB, frame_X_RXB]

##########################################################
#################### TESTS SETTINGS ######################
##########################################################

########## NOISE MESURE ##########

##### SPECTRUM ANALYZER #####
NOISE_ATTENUATION = X                          #dB
NOISE_REFERENCE_LEVEL = -X0                    #dBm
NOISE_REFERENCE_LEVEL_OFFSET = X               #dB
NOISE_START_FREQUENCY = Xe6                    #Hz
NOISE_STOP_FREQUENCY = X00e6                   #Hz
NOISE_RESOLUTION_BANDWIDTH = X0e3              #Hz
NOISE_VIDEO_BANDWIDTH = X00e3                  #Hz
NOISE_DETECTOR = "RMS"

########## TRANSFER FUNCTION MESURE ##########
```

```
##### SPECTRUM ANALYZER #####
TF_REFERENCE_LEVEL_OFFSET = X                    #dB
TF_START_FREQUENCY = Xe6                         #Hz
TF_STOP_FREQUENCY = X00e6                        #Hz
TF_RESOLUTION_BANDWIDTH = X0e3                   #Hz
TF_VIDEO_BANDWIDTH = X0e3                        #Hz
TF_DETECTOR = "POSITIVE"

##### ARBITRARY WAVEFORM GENERATOR #####
TF_PATH_AND_FILE_CHIRP = "Z:/signals/chirp/XXXXXXX.awg"   #
    file with extension .awg
TF_SAMPLING_RATE_AWG    = X.2e9                          #Hz

########## MTPR MESURE ##########

##### SPECTRUM ANALYZER #####
MTPR_REFERENCE_LEVEL_OFFSET = X                  #dB
MTPR_CENTER_FREQUENCY_PREVIOUS = Xe6             #Hz
MTPR_SPAN_PREVIOUS = X00e3                       #Hz
MTPR_RESOLUTION_BANDWIDTH_PREVIOUS = X00         #Hz
MTPR_START_FREQUENCY = Xe6                       #Hz
MTPR_STOP_FREQUENCY = X0e6                       #Hz
MTPR_RESOLUTION_BANDWIDTH = X00                  #Hz
MTPR_VIDEO_BANDWIDTH = Xe3                       #Hz
MTPR_DETECTOR = "POSITIVE"
MTPR_NUM_CARRIERS = X5
MTPR_NUM_NOTCHES = X

##### ARBITRARY WAVEFORM GENERATOR #####
MTPR_PATH_AND_FILE_OFDM = "Z:/signals/OFDM/XXXXXXXX/
    X_XX_XXXXXX_XXX_XXXXX.awg"   #file with extension .awg
MTPR_OUTPUT_VOLTAGE_AWG   = X                          #V
MTPR_SAMPLING_RATE_AWG   = X00e6                       #Hz
```

```
########## STABILITY MESURE ##########

##### DIGITAL OSCILLOSCOPE #####
STB_NUM_AVERAGES = X6
STB_VERTICAL_SCALE_1 = X0e-3                    #V/div
STB_VERTICAL_SCALE_2 = X0e-3
STB_HORIZONTAL_SCALE = X00e-9                   #s/div

##### ARBITRARY WAVEFORM GENERATOR #####
STB_PATH_AND_FILE_SQUARE = "Z:/signals/OFDM/XXXXXXXX/
    XXXXXX.awg"   #file with extension .awg
STB_OUTPUT_VOLTAGE_AWG  = X.1                                   #V
STB_SAMPLING_RATE_AWG   = Xe9                                   #Hz
```

Apéndices C

Librerías de control remoto de instrumentos

C.1. Introducción

La librería de cada uno de los instrumentos en un set de comandos SCPI predefinidos que a bajo nivel hacen llamadas a la librería **PyVISA** mediante las funciones write y read. Debido a que los recuros de Robot Framework son ficheros ASCII, vamos a emplear la nomenclatura siguiente: return = FUNCTION (args).

En esencia, la librería PyVISA realiza los siguientes pasos:

1. visa_id = visa.instrument('TCPIP::%s::%s::INSTR' % (str(host), str(device))). *host y device definidos en el fichero external_equipment.py*
2. visa_id.write(str(command_string))
3. values = visa_id.read()

C.2. Analizador de espectro

id = OPEN SESSION SA (instr)
instr: instrumento físico definido en el fichero analog_bench.py.
id: manejador (handler) para hacer referencia al instrumento en cualquier comando.
Envía *CLS, *RST e *IDN?.

void = CLOSE SESSION SA (id)
Libera el manejador id.
Establece la atenuación del analizador al máximo para protegerlo.

void = SET INPUT ATTENUATION (id, att_dB)

void = SET REFERENCE LEVEL (id, reflevel_dBm)

void = SET FREQUENCY RANGE (id, start_freq_Hz, stop_freq_Hz)

void = SET CENTER FREQUENCY AND SPAN (id, center_freq_Hz, span_Hz)

void = SET RESOLUTION BANDWIDTH (id, resol_bw_Hz)

void = SET VIDEO BANDWIDTH (id, video_bw_Hz)

void = SET SWEEP TIME (id, sweep_time_s)

void = SET REFERENCE LEVEL OFFSET (id, reflevel_offset_dB)

void = SET DETECTOR (id, detector)

[freq, power] = OBTAIN SPECTRUM ANALYZER MEASURE THROUGH CENTER FREQUENCY AND SPAN (id, start_freq_Hz, stop_freq_Hz, att_dB, reflevel_dBm, resol_bw_Hz, video_bw_Hz, sweep_time_s)

C.3. Data logger

id = OPEN SESSION DL (instr)

void = CLOSE SESSION DL (id)

volt = GET VDC CH1:CH20 (id)

curr = GET IDC CH21,CH22 (id)

C.4. Fuente de alimentación

id = OPEN SESSION PS (instr)

void = CLOSE SESSION PS (id)
Libera el manejador.
OUTPUT OFF.

void = SET VOLTAGE PS (id, source, volt_V)

void = SET OUTPUT ON PS (id)

C.5. Generador de señales arbitrarias

id = OPEN SESSION AWG (instr)

void = CLOSE SESSION AWG (id)
Libera el manejador.
OUTPUT OFF.

void = CHANGE WORK DIRECTORY AND LOAD FILE (id, path_with_filename)

void = RUN/STOP AWG (id, RUN_STOP)

void = SET OUTPUT AWG (id, ON_OFF)

void = SET OUTPUT VOLTAGE (id, volt_V)

void = SET SAMPLING FREQUENCY (id, freq_samp_Hz)

C.6. Máquina de temperatura

id = OPEN SESSION TFS (instr)

void = CLOSE SESSION TFS (id)
Libera el manejador.
Deshabilita aire.

void = SET TEMPERATURE (id, temp_C)

void = SET SOAKTIME (id, soaktime_s)

valid_temp = CHECK TEMPERATURE (id, temp_C, tolerance_C soaktime_s)
Comprueba si el sensor ha alcanzado la temperatura y devuelve True o False.

C.7. Osciloscopio digital

id = OPEN SESSION DO (instr)

void = CLOSE SESSION DO (id)

void = SET VERTICAL AND HORIZONTAL SCALE (id, v_scale_V/div, h_scale_s/div)

void = SAVE SNAPSHOT (id)
Almacena la captura de pantalla del osciloscopio en el directorio establecido en external_equipment.py

Apéndices D

Códigos del módulo SPI master

D.1. Introducción

En este apéndice se muestra el código de programación del microcontrolador (módulo arduino SPI master) y el código del sender de tramas por puerto serie.

D.2. Código del microcontrolador (SPI_microcontrolador.ino)

```
#include <SPI.h>

const byte SSpin = 10;
byte frame2tx[4];
byte framerx[4];

void setup() {

  Serial.begin(9600);
  pinMode (SSpin, OUTPUT);
  SPI.begin();
  SPI.setBitOrder(MSBFIRST);
```

```
  SPI.setDataMode(SPI_MODE3);
  SPI.setClockDivider(SPI_CLOCK_DIV128);
  digitalWrite(SSpin,HIGH);
}

void loop() {

  if(Serial.available() >= 4){
    for(int i=0; i<4; i++){
      frame2tx[i] = Serial.read();
    }

        digitalWrite(SSpin,LOW);
        delay(0.5);

        for(int i=0; i<4; i++){
                framerx[i] = SPI.transfer(frame2tx[i]);
                delay(0.75);
        }

        delay(0.5);
        digitalWrite(SSpin,HIGH);

        for(int i=0; i<4; i++){
                Serial.write(framerx[i]);
                delay(0.75);
        }

        Serial.end();

  }
}
```

D.3. Código del sender (SPI_sender.py)

```
#!/usr/bin/env python
# -*- coding: iso-8859-15 -*-

#---------------------------------------------------
# Python script to transmit
# and receive through serial port SPI frames.
#---------------------------------------------------

import sys
import serial


#-- Number of port
#-- To do multiplatform this script,
#-- you have to use numbers in range [0:255]
#-- On linux use string like /dev/ttyUSBx
Port = "/dev/ttyACM0"
Baud = 9600

#-- Strings to send

SPIframes = sys.argv[1:]
frames2tx = [];

for frame in SPIframes:
  frames2tx.append(frame.decode("hex"));

#---------------------------------------------------
#-- Open serial port.
#---------------------------------------------------
try:
```

```
    s = serial.Serial(Port, Baud)
    s.timeout=1;

except serial.SerialException:
    #-- Error opening serial port
    sys.stderr.write("Error opening serial port (%s)\n" %
        str(Port))
    sys.exit(1)

#-----------------------------------------------------
#-- Send and receive SPI frames
#-----------------------------------------------------

for frame in frames2tx:
    s.write(frame);

framesrx = [];
for i in range (len(frames2tx)):
    framesrx.append([hex(ord(z)) for z in list(s.read(size
        =4))]);

#-- Close serial port
s.close()

print framesrx;
```

Apéndices E

Códigos de la GUI

E.1. Introducción

En este apéndice se muestra el código fuente de la GUI y el código principal de Robot Framework para poder realizar la ejecución desde la GUI.

E.2. Código de la GUI (Remote_control_GUI.py)

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import sys
import os
import re
import time
import datetime as dt
import pango
import signal
import gobject
import threading
import locale
```

```
import pygtk
pygtk.require("2.0")
import gtk
import gtk.glade
import pickle
import visa
import paramiko
import subprocess

class Widgets:
  def __init__(self):
      self.ui = """CODIGO_XML_GENERADO_CON_GLADE"""
      self.widgets = gtk.glade.xml_new_from_buffer(self.ui
          ,len(self.ui))

  def __getitem__(self,key):
      return self.widgets.get_widget(key)


class Remote_Control_GUI:
  """Remote_control_GUI_by_Rafa_Perez"""
  def __init__(self):
      #Set the UI
      self.widgets = Widgets()

      #Connect widget with signals handlers
      connections = {
        'window1/destroy'                : self.
          onDestroyWindow,
        'button5/clicked'                : self.
          onClickedPowerSupply,
        'button7/clicked'                : self.
          onClickedPowerSupplyOFF,
```

```
            'button1/clicked'                    : self.
                onClickedSPIFrames,
            'button8/clicked'                    : self.
                onClickedAWGSettings,
            'button2/clicked'                    : self.
                onClickedAWGSettingsOFF,
            'button3/clicked'                    : self.
                onClickedSASettings,
            'button4/clicked'                    : self.
                onClickedTests,
            'button9/clicked'                    : self.
                onClickedKill,
            }

        for wid_con, func in connections.iteritems():
            wid,con = wid_con.split('/')
            self.widgets[wid].connect(con,func)


        self.widgets["window1"].show()

    ########        Events Handlers          ########

    def onClickedPowerSupply(self, widget):
        volt1 = self.widgets["entry18"].get_text()
        volt2 = self.widgets["entry20"].get_text()

        PS_id = visa.instrument('TCPIP::%s::%s::INSTR' %("X
            .X.X.X", "gpib0,X"))
        PS_id.write("""INSTRUMENT OUTPUT1""")
        PS_id.write("""VOLTAGE %V""" %str(volt1))
        PS_id.write("""INSTRUMENT OUTPUT2""")
        PS_id.write("""VOLTAGE %V""" %str(volt2))
```

```
        PS_id.write("""OUTPUT_ON""")

    def onClickedPowerSupplyOFF(self, widget):
        PS_id = visa.instrument('TCPIP::%s::%s::INSTR' %("X
            .X.X.X", "gpib0,X"))
        PS_id.write("""OUTPUT_OFF""")

    def onClickedSPIFrames(self, widget):
        frame1 = self.widgets["entry1"].get_text()
        frame2 = self.widgets["entry2"].get_text()
        frame3 = self.widgets["entry3"].get_text()
        frame4 = self.widgets["entry4"].get_text()

        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.
            AutoAddPolicy())
        ssh.connect('X.X.X.X', username='XXXX', password='
            XXXX')

        if len(frame1)>0:
            if len(frame1)==8:
                stdin, stdout, stderr = ssh.exec_command("""
                    sudo python SPI-sender.py "%s" """ %str(
                    frame1))
                spiout = stdout.readlines()
                spiout = "".join(["%02x"%(int(x,16)) for x
                    in spiout[0][1:-2].replace("[","").replace
                    ("]","").replace("'","").split(",")]).
                    upper()
                self.widgets["entry15"].set_text("")
                self.refresh()
                time.sleep(0.3)
                self.widgets["entry15"].set_text(spiout)
```

```
            self.refresh()
        else:
            self.showErrorMessage("You must insert a
                valid SPI FRAME 1")

    if len(frame2)>0:
        if len(frame2)==8:
            stdin, stdout, stderr = ssh.exec_command("""
                sudo python SPI-sender.py "%s" """ %str(
                frame2))
            spiout = stdout.readlines()
            spiout = "".join(["%02x"%(int(x,16)) for x
                in spiout[0][1:-2].replace("[","").replace
                ("]","").replace("'","").split(",")]).
                upper()
            self.widgets["entry15"].set_text("")
            self.refresh()
            time.sleep(0.3)
            self.widgets["entry15"].set_text(spiout)
            self.refresh()
        else:
            self.showErrorMessage("You must insert a
                valid SPI FRAME 2")

    if len(frame3)>0:
        if len(frame3)==8:
            stdin, stdout, stderr = ssh.exec_command("""
                sudo python SPI-sender.py "%s" """ %str(
                frame3))
            spiout = stdout.readlines()
```

```
            spiout = "".join(["%02x"%(int(x,16)) for x
                in spiout[0][1:-2].replace("[","").replace
                ("]","").replace("'","").split(",")]).
                upper()
            self.widgets["entry15"].set_text("")
            self.refresh()
            time.sleep(0.3)
            self.widgets["entry15"].set_text(spiout)
            self.refresh()
        else:
            self.showErrorMessage("You must insert a
                valid SPI_FRAME 3")

    if len(frame4)>0:
        if len(frame4)==8:
            stdin, stdout, stderr = ssh.exec_command("""
                sudo python SPI-sender.py "%s" """ %str(
                frame4))
            spiout = stdout.readlines()
            spiout = "".join(["%02x"%(int(x,16)) for x
                in spiout[0][1:-2].replace("[","").replace
                ("]","").replace("'","").split(",")]).
                upper()
            self.widgets["entry15"].set_text("")
            self.refresh()
            time.sleep(0.3)
            self.widgets["entry15"].set_text(spiout)
            self.refresh()
        else:
            self.showErrorMessage("You must insert a
                valid SPI_FRAME 4")

    ssh.close()
```

```
def onClickedAWGSettings(self, widget):

    path,signal= os.path.split(self.widgets["entry6"].
        get_text())
    out_volt_1 = self.widgets["entry7"].get_text()
    out_volt_2 = self.widgets["entry21"].get_text()
    sample_rate = self.widgets["entry8"].get_text()

    AWG_id = visa.instrument('TCPIP::%s::%s::INSTR' %("
        X.X.X.X", "inst0"))
    AWG_id.write("""MMEMORY:CDIRECTORY "%s" """ %str(
        path))
    AWG_id.write("""AWGCONTROL:SRESTORE "%s" """ %str(
        signal))
    AWG_id.write("""SOURCE1:VOLTAGE %sV""" %str(
        out_volt_1))
    AWG_id.write("""SOURCE2:VOLTAGE %sV""" %str(
        out_volt_2))
    AWG_id.write("""SOURCE:FREQUENCY %sHz""" %str(
        sample_rate))
    AWG_id.write("""AWGCONTROL:RUN""")
    AWG_id.write("""OUTPUT1 ON""")
    AWG_id.write("""OUTPUT2 ON""")

def onClickedAWGSettingsOFF(self, widget):
    AWG_id = visa.instrument('TCPIP::%s::%s::INSTR' %("
        X.X.X.X", "inst0"))
    AWG_id.write("""AWGCONTROL:STOP""")
    AWG_id.write("""OUTPUT1 OFF""")
    AWG_id.write("""OUTPUT2 OFF""")
```

```
def onClickedSASettings(self, widget):
    att = self.widgets["entry9"].get_text()
    ref_level = self.widgets["entry10"].get_text()
    start_freq = self.widgets["entry11"].get_text()
    stop_freq = self.widgets["entry12"].get_text()
    resol_bw = self.widgets["entry13"].get_text()
    video_bw_auto = self.widgets["combobox2"].
       get_active_text()
    video_bw = self.widgets["entry14"].get_text()
    det = self.widgets["combobox1"].get_active_text()
    ref_level_off = self.widgets["entry16"].get_text()

    SA_id = visa.instrument('TCPIP::%s::%s::INSTR' %("X
       .X.X.X", "inst0"))
    SA_id.write("""INPUT:ATTENUATION %dB""" %str(att))
    SA_id.write("""DISPLAY:TRACE:Y:RLEVEL %dBm""" %str
       (ref_level))
    SA_id.write("""SENSE:FREQUENCY:START %Hz""" %str(
       start_freq))
    SA_id.write("""SENSE:FREQUENCY:STOP %Hz""" %str(
       stop_freq))
    SA_id.write("""BAND:RES %Hz""" %str(resol_bw))
    if video_bw_auto == "ON":
        SA_id.write("""BAND:VID:AUTO ON""")
    elif video_bw_auto == "OFF":
        SA_id.write("""BAND:VID %Hz""" %str(video_bw))
    else:
        pass
    SA_id.write("""DETECTOR1 %s""" %str(det))
    SA_id.write("""DISPLAY:TRACE:Y:RLEVEL:OFFS %dB"""
       %str(ref_level_off))

def onClickedTests(self, widget):
```

```
DC = self.widgets["checkbutton1"].get_active()
NoisePSD = self.widgets["checkbutton2"].get_active()
OutputTrace = self.widgets["checkbutton3"].
    get_active()
Enabled_Trace1 = self.widgets["checkbutton6"].
    get_active()
Enabled_Trace2 = self.widgets["checkbutton7"].
    get_active()
Enabled_Trace3 = self.widgets["checkbutton8"].
    get_active()
MTPR = self.widgets["checkbutton4"].get_active()
STB = self.widgets["checkbutton5"].get_active()

temp = self.widgets["entry17"].get_text()
soak = self.widgets["entry22"].get_text()
path = self.widgets["entry19"].get_text()
name = self.widgets["entry5"].get_text()

file_settings_ROBOT = open("GUI_settings.py","w")
if len(temp)>0 and -XX0<float(temp)<X30:
    file_settings_ROBOT.write("ENABLE_TEMPERATURE = 
        1\n")
    file_settings_ROBOT.write("TEMP = %s\n" %str(
        temp))

else:
    file_settings_ROBOT.write("ENABLE_TEMPERATURE = 
        0\n")

file_settings_ROBOT.write("""TEST2RUN = [%s,%s,%s,%s
    ,%s]\n""" %(str(DC),str(NoisePSD),str(OutputTrace)
    ,str(MTPR),str(STB)))
```

```
        file_settings_ROBOT.write("""ENABLED_TRACES = [%s,%s
            ,%s]\n""" %(str(Enabled_Trace1),str(Enabled_Trace2
            ),str(Enabled_Trace3)))
        file_settings_ROBOT.write("""FILENAME = "%s"\n""" %
            str(name))
        file_settings_ROBOT.write("""SOAK = "%s"\n""" %str(
            soak))
        file_settings_ROBOT.close()

        global proc
        proc = subprocess.Popen(["""pybot --loglevel NONE --
            variablefile XXXX/benches/analog_bench.py --
            variablefile XXXX/benches/settings_dvt.py --
            variablefile GUI_settings.py --outputdir %s --test
            Tests.Analogic.GUI-DVT.GUI-DVT_tests/""" %str(
            path)], shell=True)

    def onClickedKill(self, widget):
        proc.terminate()

    def showErrorMessage(self, data=None):
        message = gtk.MessageDialog(None, gtk.DIALOG_MODAL,
            gtk.MESSAGE_INFO, gtk.BUTTONS_CLOSE, data)
        message.show()
        resp = message.run()
        if resp == gtk.RESPONSE_CLOSE:
            message.destroy()

    def onDestroyWindow(self, widget):
        gtk.main_quit()

    ########      Auxiliary Functions      ########
```

```
  def refresh(self):
      while gtk.events_pending():
          gtk.main_iteration()


if __name__ == "__main__":
    gobject.threads_init()
    gtk.gdk.threads_init()

    encoding = locale.getpreferredencoding()
    utf8conv = lambda x : unicode(x, encoding).encode('
       utf8')

    Remote_Control_GUI()

    signal.signal(signal.SIGINT, signal.SIG_DFL) # ^C
       exits the application

##  print "Before gtk.main\n"
    gtk.gdk.threads_enter()
    gtk.main()
    gtk.gdk.threads_leave()
```

E.3. Código de Robot Framework (GUI-DVT.txt)

```
*** Settings ***
Library           Dialogs
Library           libs/libraries/testlibraries/
   TransferFunction.py
Library           libs/libraries/testlibraries/
   HtmlReportLibrary.py
Resource          Multi_Tone_Power_Ratio.txt
```

```
Resource          Noise_Power_Spectrum_Density.txt
Resource          libs/resources/
    ArbitraryWaveformGeneratorVisaResource.txt
Resource          libs/resources/PowerSupplyVisaResource.
    txt
Resource          libs/resources/
    TemperatureForceSystemVisaResource.txt
Resource          libs/resources/DataLoggerVisaResource.
    txt
Resource          libs/resources/
    DigitalOscilloscopeVisaResource.txt
Resource          libs/resources/
    SpectrumAnalyzerVisaResource.txt
Resource          libs/resources/SPIResource.txt

*** Test Cases ***
GUI-DVT
    [Setup]
    Run Keyword If    ${ENABLE_TEMPERATURE} == 1    GUI
        Tests with TFS    ELSE    GUI Tests
    [Teardown]

*** Keywords ***
GUI Tests with TFS
    ${id}    ${name}=    Open session TFS    ${
        TEMPERATURE_FS[0]}
    Set temperature    ${id}    1    ${TEMP}
    Set temperature sensor    ${id}    1
    Set soaktime    ${id}    1    ${SOAK}
    Start process    ${id}    1
    ${valid_temp}=    Check temperature    ${id}    ${TEMP
        }    2    ${SOAK}
    Run Keyword If    ${valid_temp} == True    GUI Tests
```

```
    Close session TFS    ${id}
    ####################

GUI Tests
    Run Keyword If    ${TEST2RUN[0]} == True    GUI
      TEMPLATE - Test Vdc and Idc    ${FILENAME}
    Run Keyword If    ${TEST2RUN[1]} == True    GUI
      TEMPLATE - Test Noise PSD    ${FILENAME}
    Run Keyword If    ${TEST2RUN[2]} == True    GUI
      TEMPLATE - Test Output Trace    ${FILENAME}
    Run Keyword If    ${TEST2RUN[3]} == True    GUI
      TEMPLATE - Test MTPR    ${FILENAME}
    Run Keyword If    ${TEST2RUN[4]} == True    GUI
      TEMPLATE - Test Stability    ${FILENAME}

GUI TEMPLATE - Test Vdc and Idc
    [Arguments]    ${filename}
    ${id}    ${name}=    Open session DL    ${DATA_LOGGERS
      [1]}
    ${measures_Vdc}=    Get Vdc @ CH101:CH120    ${id}
    ${measures_Idc}=    Get Idc @ CH121:CH122    ${id}
    Close session DL    ${id}
    # Generate data file
    ${filename}=    Set Variable    ${filename}
    : FOR    ${v}    IN RANGE    0    20
    \    OperatingSystem.Append To File    ${OUTPUTDIR}/${
      filename}_Vdc_Idc.meas    Vdc${v+1}\t\t${
      measures_Vdc[${v}]}\n
    OperatingSystem.Append To File    ${OUTPUTDIR}/${
      filename}_Vdc_Idc.meas    Idc21\t\t${measures_Idc
      [0]}\n
```

```
    OperatingSystem.Append To File    ${OUTPUTDIR}/${
       filename}_Vdc_Idc.meas    Idc22\t\t${measures_Idc
       [1]}\n

GUI TEMPLATE - Test Noise PSD
    [Arguments]    ${filename}
    ${spectrum_analyzer}=    Set Variable    ${
       SPECTRUM_ANALYZERS[1]}
    ${id}=    Open Visa session    ${spectrum_analyzer["ip
       "]}    ${spectrum_analyzer["inst"]}
    # Swith on analyzer screen (for debugging)
    Visa write    ${id}    SYSTEM:DISPLAY:UPD ON
    # Set SWEET TIME automatic
    Visa write    ${id}    SWEEP:TIME:AUTO ON
    # Calculate the sweep time
    Visa write    ${id}    SWEEP:POINTS 501
    Visa write    ${id}    INIT:CONT OFF
    Visa write    ${id}    INIT
    Visa write    ${id}    SWEEP:TIME?
    ${sweep_time}=    Visa read    ${id}
    ${sleep_time}=    Evaluate    ${sweep_time}+0.5*${
       sweep_time}
    Sleep    ${sleep_time}
    # Set the output format to ASCII and retrieve the
       stored information
    Visa write    ${id}    :FORM ASC; :TRAC? TRACE1
    ${output}=    Visa read    ${id}
    @{data_points}=    Split String    ${output}    ,
    # Correct the just obtained data points according to
       the used bandwidth resolution
    Visa write    ${id}    BAND?
    ${resolution_bandwidth}=    Visa read    ${id}
```

```
    ${correction}=    Calculate Correcion Factor    ${
       resolution_bandwidth}
    ${data_points}=    Evaluate    [float(x)-float(${
       correction}) for x in @{data_points}]
    # Obtain the frequencies corresponding to each of the
       data points
    Visa write    ${id}    SENS:FREQ:START?
    ${start_frequency}=    Visa read    ${id}
    Visa write    ${id}    SENS:FREQ:STOP?
    ${stop_frequency}=    Visa read    ${id}
    ${step}=    Evaluate    ( ${stop_frequency} - ${
       start_frequency} ) / len(${data_points})
    ${frequencies}=    Evaluate    [(${start_frequency} +
       ${step} * i)/1000.0 for i in range(len(${data_points
       }))]
    # Plot image
    Plot Noise PSD Trace    ${frequencies}    ${
       data_points}    ${OUTPUTDIR}/${filename}_NoisePSD.
       png    NOISE PSD
    Log Image    ${OUTPUTDIR}/${filename}_NoisePSD.png
    # Generate data file
    ${length}=    Get Length    ${frequencies}
    : FOR    ${i}    IN RANGE    ${length}
    \    OperatingSystem.Append To File    ${OUTPUTDIR}/${
       filename}_NoisePSD.meas    ${frequencies[${i}]}\t\t${
       data_points[${i}]}\n

GUI TEMPLATE - Test Output Trace
    [Arguments]    ${filename}
    ${spectrum_analyzer}=    Set Variable    ${
       SPECTRUM_ANALYZERS[1]}
    ${id}=    Open Visa session    ${spectrum_analyzer["ip
       "]}    ${spectrum_analyzer["inst"]}
```

```
# Swith on analyzer screen (for debugging)
Visa write    ${id}    SYSTEM:DISPLAY:UPD ON
# Set SWEEP TIME automatic
Visa write    ${id}    SWEEP:TIME:AUTO ON
# Set SINGLE SWEEP MODE and calculate the sweep time
   and sleep long enough for one sweep to continue
Visa write    ${id}    SWEEP:POINTS 501
${v_aux}=    Evaluate    [0 for x in range(0,501)]
# Set the output format to ASCII and retrieve the
   stored information - 1
Run Keyword If    ${ENABLED_TRACES[0]} == 1    Visa
   write    ${id}    FORMat ASCii; TRAC? TRACE1
${output}=    Run Keyword If    ${ENABLED_TRACES[0]}
   == 1    Visa read    ${id}
${data_points1}=    Run Keyword If    ${ENABLED_TRACES
   [0]} == 1    Split String    ${output}    ,    ELSE
...    Set Variable    ${v_aux}
# Set the output format to ASCII and retrieve the
   stored information - 2
Run keyword if    ${ENABLED_TRACES[1]} == 1    Visa
   write    ${id}    FORMat ASCii; TRAC? TRACE2
${output}=    Run Keyword If    ${ENABLED_TRACES[1]}
   == 1    Visa read    ${id}    ELSE    Set Variable
...    ${v_aux}
${data_points2}=    Run Keyword If    ${ENABLED_TRACES
   [1]} == 1    Split String    ${output}    ,    ELSE
...    Set Variable    ${v_aux}
# Set the output format to ASCII and retrieve the
   stored information - 3
Run keyword if    ${ENABLED_TRACES[2]} == 1    Visa
   write    ${id}    FORMat ASCii; TRAC? TRACE3
${output}=    Run keyword if    ${ENABLED_TRACES[2]}
   == 1    Visa read    ${id}    ELSE    Set Variable
```

```
    ...     ${v_aux}
    ${data_points3}=    Run keyword if    ${ENABLED_TRACES
       [2]} == 1    Split String    ${output}    ,    ELSE
    ...     Set Variable    ${v_aux}
    # Obtain the frequencies corresponding to each of the
       data points
    Visa write    ${id}    SENS:FREQ:START?
    ${start_frequency}=    Visa read    ${id}
    Visa write    ${id}    SENS:FREQ:STOP?
    ${stop_frequency}=    VXI11 read    ${id}
    ${step}=    Evaluate    (${stop_frequency} - ${
       start_frequency}) / len(${v_aux})
    ${frequencies}=    Evaluate    [(${start_frequency} +
       ${step} * i)/1000.0 for i in range(len(${v_aux}))]
    ${data_points}=    Create List    ${data_points1}    $
       {data_points2}    ${data_points3}
    ###########
    Plot TF Trace    ${frequencies}    ${data_points}    $
       {OUTPUTDIR}/${filename}_OutputTrace.png    OUTPUT
       TRACES
    Log Image    ${OUTPUTDIR}/${filename}_OutputTrace.png
    # Generate data file
    ${length}=    Get Length    ${frequencies}
    : FOR    ${i}    IN RANGE    ${length}
    \    OperatingSystem.Append To File    ${OUTPUTDIR}/${
       filename}_OutputTrace.meas    ${frequencies[${i}]}\t
       \t${data_points[0][${i}]}\t\t${data_points[1][${i
       }]}\t\t${data_points[2][${i}]}\n

GUI TEMPLATE - Test MTPR
    [Arguments]    ${filename}
    ${spectrum_analyzer}=    Set Variable    ${
       SPECTRUM_ANALYZERS[1]}
```

```
${id}=    Open Visa session    ${spectrum_analyzer["ip
   "]}    ${spectrum_analyzer["inst"]}
Visa write    ${id}    INPUT:ATTENUATION?
${att}=    Visa read    ${id}
Visa write    ${id}    DISPLAY:TRACE:Y:RLEVEL?
${ref_level}=    Visa read    ${id}
Visa write    ${id}    BAND:RES?
${resol_BW}=    Visa read    ${id}
Visa write    ${id}    BAND:VID?
${video_BW}=    Visa read    ${id}
Visa write    ${id}    DISPLAY:TRACE:Y:RLEVEL:OFFS?
${ref_level_offset}=    Visa read    ${id}
Visa write    ${id}    DETECTOR?
${detector}=    Visa read    ${id}
# Obtain frequency spacing and first carrier frequency
Set configuration for MTPR    ${id}    ${att}    ${
   ref_level}    ${MTPR_CENTER_FREQUENCY_PREVIOUS}    $
   {MTPR_SPAN_PREVIOUS}    ${
   MTPR_RESOLUTION_BANDWIDTH_PREVIOUS}
...    ${MTPR_VIDEO_BANDWIDTH}    ${
   MTPR_REFERENCE_LEVEL_OFFSET}
${freq_spacing}    ${first_carrier_frequency}=
   Obtain frequency spacing and first carrier frequency
       ${id}    ${detector}
${span}=    Evaluate    ((${MTPR_NUM_NOTCHES}+4)*(${
   freq_spacing}/1000))*1E6    # Value in Hz
${max_iter}=    Evaluate    (${MTPR_STOP_FREQUENCY}-${
   MTPR_START_FREQUENCY})/(${freq_spacing}*1000)/(${
   MTPR_NUM_CARRIERS}+${MTPR_NUM_NOTCHES})+1    # All
   frequencies must be in Hz
${max_iter}=    Integer Part    ${max_iter}
# Calculate MTPR
${mtpr}=    Create List
```

```
${freq}=    Create List
# First notch
${center_frequency_first_notch}=    Evaluate    ((${
   first_carrier_frequency}+${freq_spacing}*(${
   MTPR_NUM_CARRIERS}+${MTPR_NUM_NOTCHES}/2.0-0.5))
   /1000)*1E6    # Value in Hz
Set configuration for MTPR    ${id}    ${att}    ${
   ref_level}    ${center_frequency_first_notch}    ${
   span}    ${resol_BW}
...    ${video_BW}    ${ref_level_offset}
${values}    ${max_power_carrier}=    Obtain MTPR
   trace and calculate    ${id}    ${detector}
Append To List    ${freq}    ${values[0]}
Append To List    ${mtpr}    ${values[1]}
# Following notches
${m_factor}=    Evaluate    ${freq_spacing}/1000*(${
   MTPR_NUM_NOTCHES}+${MTPR_NUM_CARRIERS})    # y = mx
   + n
${m_factor}=    Trucate Number    ${m_factor}
${m_factor}=    Evaluate    ${m_factor}*1E6    # Value
    in Hz
: FOR    ${iter}    IN RANGE    1    ${max_iter}
\    ${center_frequency}=    Evaluate    ${
   center_frequency_first_notch}+${m_factor}*${iter}
      # cf = m*iter + cffn    # Value in Hz
\    Set center frequency and span    ${id}    ${
   center_frequency}    ${span}
\    ${values}    ${new_max_power_carrier}=    Obtain
   MTPR trace and calculate    ${id}    ${detector}
\    ${diff}=    Evaluate    ${new_max_power_carrier}-
   ${max_power_carrier}
```

```
    \    Run Keyword If    ${diff} < -10    Get down
       reference level and attenuation    ${id}    ${diff}
          ${ref_level}
    \    ...    ${att}
    \    Run Keyword If    ${diff} > +10    Get up
       reference level and attenuation    ${id}    ${diff}
          ${ref_level}
    \    ...    ${att}
    \    Append To List    ${freq}    ${values[0]}
    \    Append To List    ${mtpr}    ${values[1]}
    \    ${max_power_carrier}=    Set Variable    ${
       new_max_power_carrier}
    Plot MTPR Trace    ${freq}    ${mtpr}    ${OUTPUTDIR}/
       ${filename}_MTPR.png    MTPR
    Log Image    ${OUTPUTDIR}/${filename}_MTPR.png
    # Generate data file
    ${length}=    Get Length    ${freq}
    : FOR    ${i}    IN RANGE    ${length}
    \    OperatingSystem.Append To File    ${OUTPUTDIR}/${
       filename}_MTPR.meas    ${freq[${i}]}\t\t${mtpr[${i
       }]}\n

GUI TEMPLATE - Test Stability
    [Arguments]    ${filename}
    # Configure AWG with signal to Transfer Function
       mesure
    ${id}    ${name}=    Open session AWG    ${
       ARBITRARY_WAVEFORM_GENERATORS[0]}
    Change working directory and load file    ${id}    ${
       STB_PATH_AND_FILE_SQUARE}
    Set configuration AWG    ${id}    1    ${
       output_voltage_AWG}    ${STB_SAMPLING_RATE_AWG}
```

```
Set configuration AWG    ${id}    2    ${
   output_voltage_AWG}    ${STB_SAMPLING_RATE_AWG}
RUN/STOP AWG    ${id}    RUN
Enable/Disable Output AWG    ${id}    1    ON
Enable/Disable Output AWG    ${id}    2    ON
# Obtain trace
${id2}    ${name2}=    Open session DO    ${
   DIGITAL_OSCILLOSCOPES[0]}
Set acquisition mode average    ${id2}    ${
   STB_NUM_AVERAGES}
Set Vertical and Horizontal Scale    ${id2}    1    ${
   STB_VERTICAL_SCALE_1}    ${STB_HORIZONTAL_SCALE}
Set Vertical and Horizontal Scale    ${id2}    2    ${
   STB_VERTICAL_SCALE_2}    ${STB_HORIZONTAL_SCALE}
Set channel offset    ${id2}    1    1.15
Set channel offset    ${id2}    2    1.18
Get fall time    ${id2}
Get rise time    ${id2}
Get positive overshoot    ${id2}
Get negative overshoot    ${id2}
Sleep    2
Save snapshot    ${id2}    ${filename}_STB.png
Close session DO    ${id2}
# Close session AWG
Close session AWG    ${id}
# Generate setup file
${filename}=    Set Variable    ${dut}_${time}_${
   channel}_${temp}_${volt}_${SPI_11}_${SPI_01}_${
   SPI_10}_${SPI_00}
OperatingSystem.Append To File    ${filename}_STB.
   setup    STABILITY\n
OperatingSystem.Append To File    ${filename}_STB.
   setup    ARBITRARY WAVEFORM GENERATOR\n
```

```
    OperatingSystem.Append To File    ${filename}_STB.
       setup    "${name}"\n
    OperatingSystem.Append To File    ${filename}_STB.
       setup    SIGNAL = ${STB_PATH_AND_FILE_SQUARE}\n
    OperatingSystem.Append To File    ${filename}_STB.
       setup    OUTPUT VOLTAGE = ${STB_OUTPUT_VOLTAGE_AWG}V
       \n
    OperatingSystem.Append To File    ${filename}_STB.
       setup    SAMPLING RATE = ${STB_SAMPLING_RATE_AWG}Hz\
       n
    OperatingSystem.Append To File    ${filename}_STB.
       setup    DIGITAL OSCILLOSCOPE\n
    OperatingSystem.Append To File    ${filename}_STB.
       setup    "${name2}"\n
    OperatingSystem.Append To File    ${filename}_STB.
       setup    NUM_AVG = ${STB_NUM_AVERAGES}\n
    OperatingSystem.Append To File    ${filename}_STB.
       setup    VERTICAL_SCALE_1 = ${STB_VERTICAL_SCALE_1}\
       n
    OperatingSystem.Append To File    ${filename}_STB.
       setup    VERTICAL_SCALE_2 = ${STB_VERTICAL_SCALE_2}\
       n
    OperatingSystem.Append To File    ${filename}_STB.
       setup    HORIZONTAL_SCALE = ${STB_HORIZONTAL_SCALE}\
       n
    OperatingSystem.Append To File    ${filename}_STB.
       setup    ${SPI_11}_${SPI_01}_${SPI_10}_${SPI_00}\n
    # Return
    ${return}=    Create List    TestSTB    ${temp}    ${
       volt5}    ${volt33}    ${Gain}
    ...    ${SPI_11}    ${SPI_10}    ${SPI_01}    ${SPI_00
       }    ${name2}    ${STB_VERTICAL_SCALE_1}
```

```
... ${STB_VERTICAL_SCALE_2} ${
    STB_HORIZONTAL_SCALE} ${STB_NUM_AVERAGES} ${
    name} ${STB_PATH_AND_FILE_CHIRP} ${
    output_voltage_AWG}
... ${STB_SAMPLING_RATE_AWG} ${filename}_STB.png
[Teardown]
```

Apéndices F

Informe final

F.1. Introducción

En este apéndice se muestra un extracto del informe final generado por el DVT. Hay que recordar que el informe final es un fichero html visualizable con cualquier explorador web y mediante el cual podemos pasarlo a pdf de una manera muy sencilla.

DVT REPORT

SAMPLE: SAMPLE_NAME

TIME: 20140611_082020

CHANNEL: TXA

STATIC MEASUREMENTS

TEMP	X5C
VDD5	XV
VDD33	3.XV

Gain (dB)	XX
SPI_11	0xXXXXXXXX
SPI_10	0xXXXXXXXX
SPI_01	0xXXXXXXXX
SPI_00	0xXXXXXXXX

PIN NAME	**VALUE (V)**
NAME_1	X.637
NAME_2	X.637
NAME_3	X.631
NAME_5	X.632
NAME_6	X.625
NAME_7	X.625
NAME_8	X.613
NAME_9	X.614
NAME_10	X.483
NAME_11	X.483
NAME_12	X.483
NAME_13	X.483
NAME_14	X.416
NAME_15	X.416
NAME_16	X.011
NAME_17	X.012
NAME_18	X.392

NAME_19	X.390
NAME_20	X.181

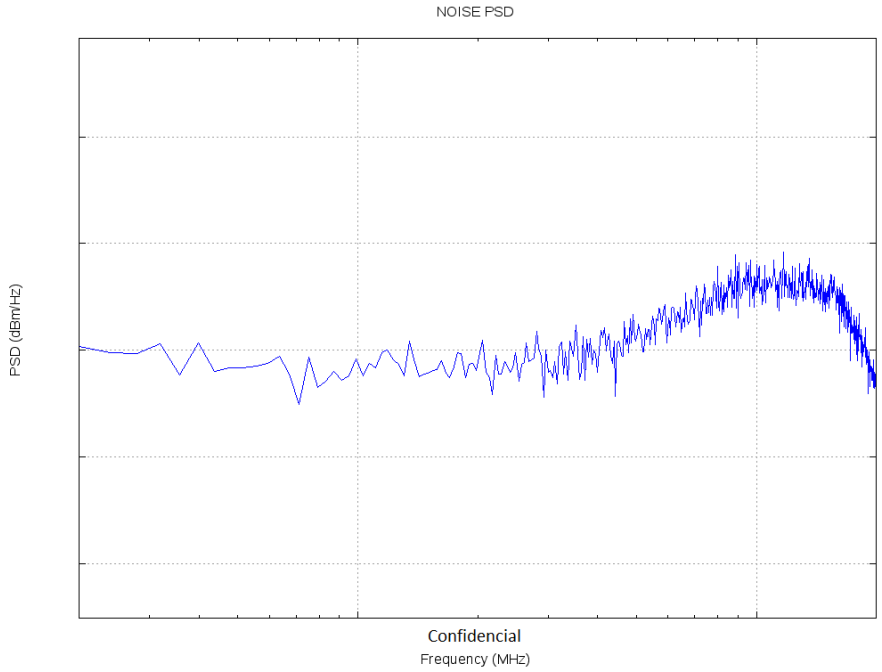
PIN NAME	VALUE (mA)
NAME_1	-X8.617
NAME_2	X37.417

NOISE PSD

TEMP	X5C
VDD5	XV
VDD33	3.XV

Gain (dB)	XX
SPI_11	0xXXXXXXXX
SPI_10	0xXXXXXXXX
SPI_01	0xXXXXXXXX
SPI_00	0xXXXXXXXX

SPECTRUM ANALYZER SETTINGS	
SP ANALYZER	XXXX,XXXXX,100017/008,4.25
ATT (dB)	X
REF LEVEL (dBm)	-X0
START FREQ (MHz)	X
STOP FREQ (MHZ)	X00
RESOL BW (kHz)	X0
VIDEO BW (kHz)	X00
REF LEVEL OFFSET (dB)	X
DETECTOR	RMS



TRANSFER FUNCTION

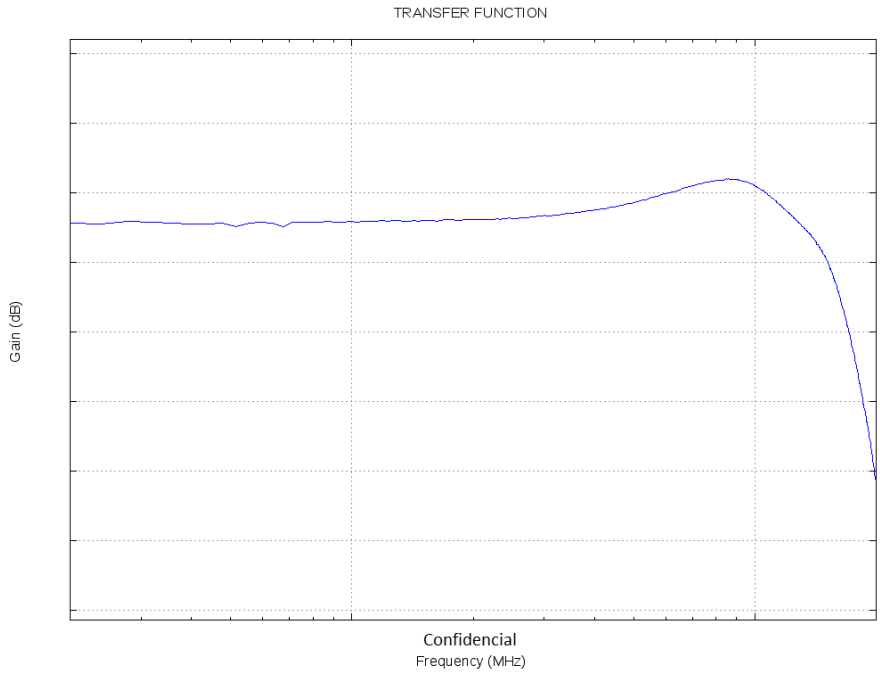
TEMP	X5C
VDD5	XV
VDD33	3.XV

Gain (dB)	XX
SPI_11	0xXXXXXXXX
SPI_10	0xXXXXXXXX
SPI_01	0xXXXXXXXX
SPI_00	0xXXXXXXXX

AWG SETTINGS	
AWG	XXXX,XXXXXXXX,B010147,SCPI:99.0 FW:4.5.0.6
SIGNAL	Z:/signals/chirp/XXXXXXX.awg
OUTPUT VOLT (Vpp)	X.050
SAMPLING RATE (GHz)	X.2

SPECTRUM ANALYZER SETTINGS	
SP ANALYZER	XXXX,XXXXX,100017/008,4.25
ATT (dB)	X0
REF LEVEL (dBm)	X
START FREQ (MHz)	X
STOP FREQ (MHZ)	X00

RESOL BW (kHz)	X0
VIDEO BW (kHz)	X0
REF LEVEL OFFSET (dB)	X
DETECTOR	POSITIVE



DYNAMIC CONSUMPTION

...

Bibliografía

[1] David Balaguer Andrés. *LOW NOISE AMPLIFIER IN SUBMICRON CMOS TECHNOLOGY FOR MARVELL G.hn INTEGRATED ANALOG FRONT END*. Proyecto Fin de Carrera, 2013. Proyecto Fin de Carrera (PFC) (ETSIT-UPV).

[2] Universidad de Alcalá. Ruido intrínseco en dispositivos electrónicos. URL http://www.depeca.uah.es/depeca/repositorio/asignaturas/32421/Ruido.pdf.

[3] Sergio Gonzalez Frasquet. *Evaluación de la calidad de la transmisión de datos de vídeo en un entorno PLC*. Proyecto Fin de Carrera, 2008. Proyecto Fin de Carrera (PFC) (ETSIT-UPV).

[4] National Instruments. Virtual instrumentation software architecture. 2014. URL http://www.ni.com/visa/. [Online; accessed 19-June-2014].

[5] José María Grima Palop. Apuntes de instrumentación electrónica. capítulo 2.

[6] Sofiène Tahar. System-on-chip design verification: Challenges and state-of-the-art, slide 18. 2012. URL http://www.mcsoc-forum.org/2012/files/MCSOC12-Tahar.pdf.

[7] Wikipedia. Ieee-488 — wikipedia, the free encyclopedia. 2014. URL http://en.wikipedia.org/w/index.php?title=IEEE-488&oldid=615781922. [Online; accessed 15-June-2014].

[8] Wikipedia. Lan extensions for instrumentation — wikipedia, the free encyclopedia. 2014. URL http://en.wikipedia.org/w/index.php?title=LAN_eXtensions_for_Instrumentation&oldid=605359885. [Online; accessed 15-June-2014].

[9] Wikipedia. Pci extensions for instrumentation — wikipedia, the free encyclopedia. 2014. URL http://en.wikipedia.org/w/index.php?title=PCI_eXtensions_for_Instrumentation&oldid=613512512. [Online; accessed 16-June-2014].

[10] Wikipedia. Serial peripheral interface bus — wikipedia, the free encyclopedia. 2014. URL http://en.wikipedia.org/w/index.php?title=Serial_Peripheral_Interface_Bus&oldid=613548396. [Online; accessed 19-June-2014].

[11] Wikipedia. Standard commands for programmable instruments — wikipedia, the free encyclopedia. 2014. URL http://en.wikipedia.org/w/index.php?title=Standard_Commands_for_Programmable_Instruments&oldid=602458170. [Online; accessed 18-June-2014].

[12] Wikipedia. Vme extensions for instrumentation — wikipedia, the free encyclopedia. 2014. URL http://en.wikipedia.org/w/index.php?title=VME_eXtensions_for_Instrumentation&oldid=613178882. [Online; accessed 17-June-2014].