

**DIRECCIÓN DE SERVICIOS DE CERTIFICACIÓN**

**LABORATORIO DE ENSAYOS DE  
COMPATIBILIDAD ELECTROMAGNÉTICA RADIADA**

**INSTALACIÓN DE LIBRERÍA VXI-11**

**SUB TÍTULO**

<b>CÓDIGO:</b> FO-IT-002		<b>N DOC:</b>	
Originado por:	Elaborado por:	Revisado por:	Aprobado por:
Br. Arias B., Jose A.	Br. Arias B., Jose A.	-	-
<b>Fecha:</b> 07/07/2017	<b>Fecha:</b> 07/07/2016	<b>Fecha:</b>	<b>Fecha:</b>

## Índice

<b>1</b>	<b>Objetivos</b>	<b>2</b>
<b>2</b>	<b>Alcance</b>	<b>2</b>
<b>3</b>	<b>Documentos de referencia</b>	<b>3</b>
3.1	Enlaces de interés . . . . .	3
<b>4</b>	<b>Términos y definiciones</b>	<b>3</b>
<b>5</b>	<b>Personal autorizado</b>	<b>3</b>
<b>6</b>	<b>Personal requerido</b>	<b>3</b>
<b>7</b>	<b>Materiales</b>	<b>3</b>
<b>8</b>	<b>Herramientas y equipos</b>	<b>3</b>
<b>9</b>	<b>Equipos de protección personal</b>	<b>3</b>
<b>10</b>	<b>Precauciones de seguridad</b>	<b>4</b>
<b>11</b>	<b>Descripción de la actividad</b>	<b>4</b>
11.1	Generalidades . . . . .	4
11.2	Librería VXI . . . . .	5
11.2.1	Archivos componentes . . . . .	6
11.2.2	Utilidades . . . . .	6
11.3	Construcción de la librería libvxi11.so . . . . .	7
<b>12</b>	<b>Anexos</b>	<b>8</b>
12.1	Script de Bash para carga automática de firmware . . . . .	8

## 1. Objetivos

- Describir el acceso a instrumentos de medición por medio del puente LAN/GPIB E5810

## 2. Alcance

En este documento se explica como establecer un puente de conexión entre instrumentos de medición GPIB y una red LAN. Para ello se establecerá la configuración del puente LAN/GPIB Agilent E5810A. Se detallará principalmente la instalación de la librería VXI-11 para linux, soporte de software que permite el acceso programático a este dispositivo.

### 3. Documentos de referencia

#### 3.1. Enlaces de interés

### 4. Términos y definiciones

Termino	Definición
VXI	VMEbus eXtensions for Instrumentation
RPC	Remote Procedure Call

### 5. Personal autorizado

Personal técnico del Cendit con interés en el acceso a instrumentos de medición GPIB por medio de un puente de redes LAN a GPIB.

### 6. Personal requerido

Ver sección 5.

### 7. Materiales

- Computador con acceso a internet.
- Puente LAN/GPIB Agilent E5810A.
- Cable ethernet cruzado.

### 8. Herramientas y equipos

Ver sección 7.

### 9. Equipos de protección personal

No se requieren equipos protección personal.

## 10. Precauciones de seguridad

Para ejecutar esta actividad no se prevén precauciones de seguridad

## 11. Descripción de la actividad

### 11.1. Generalidades

La especificación *VXI-11* fue desarrollada a comienzos de los años 90 como una parte de una especificación mayor, la del bus *VXIbus*. *VXI-11* define el protocolo de comunicaciones por medio del cual instrumentos de medición y controladores se comunican sobre una red TCP/IP.

Fue empleada en puentes LAN/GPIB antes de que existiesen los instrumentos con una interfaz LAN nativa. Un punto importante de la especificación *VXI-11* es la de lograr la interconexión de dispositivos de manera independiente de su fabricante.

Las comunicación y el paradigma de programación es similar a los soportados por los estándares IEEE-488.1 e IEEE-488.2 en el sentido de que las comunicaciones son basadas en transferencia de datos ASCII y mensajes de control IEEE-488.1, entre un controlador y un dispositivo, sobre una red de computadores. Esto se debe a que *VXI-11* fue ideada en principio para replicar ciertas características de un bus GPIB a una red LAN, incluyendo aquellas características propias del hardware (señalización del bus).

Esta especificación posee tres sub secciones, *VXI-11.1* trata la interconexión de dispositivos *VXIbus* a una red. *VXI-11.2* trata la conexión de instrumentos GPIB a una red por medio de puentes Lan a GPIB como el Agilent E5810A, *VXI-11.3* trata de la conexión de instrumentos que cumplen con el estándar IEEE-488, y que poseen una interfaz Ethernet para conexión directa a una red Lan.

Las comunicaciones dentro de la especificación *VXI-11* se efectúan sobre el protocolo ONC Remote Procedure Call (RPC). El protocolo RPC permite un enlace de tipo cliente-servidor, una aplicación (típicamente el cliente) efectúa llamadas a procedimientos en una aplicación remota (el servidor), como si estos procedimientos remotos fuesen ejecutado localmente.

RPC fue diseñado para ser independiente de un lenguaje de programación, sistema operativo o plataforma de computador en particular, el servidor RPC y el cliente RPC pueden ejecutarse en diferentes sistemas operativos y procesadores. Esta *interoperabilidad* se logra al representar los datos que viajan por la red en el formato XDR, el cual define tipos de datos estándar y el ordenamiento de los bytes de datos empleados en las llamadas RPC.

Cuando se realiza una llamada a un procedimiento RPC, los datos que se pasan a una función deben traducirse del formato del lenguaje de programación utilizado al formato XDR y en el servidor se traducen de vuelta, de XDR al formato nativo del lenguaje de programación.

Las funciones disponibles en un servidor RPC se describen por medio de un archivo RPCL (RPC Language). La definición de funciones en el archivo RPCL es muy similar a la definición

de tipos C.

En la figura se muestran las funciones RPC para uso con VXI-11 y se muestra una entrada que describe estas funciones en el archivo RPCL.



Figura 1: Puente LAN/GPIB E5810 de Agilent Technologies

El sistema operativo brinda el soporte necesario para la transferencia de datos sobre RPC, en forma de librerías. Por ejemplo la función `clnt_call()` del sistema operativo permite llamar a funciones RPC, aunque su uso es tedioso. Una forma de facilitar las llamadas RPC es por medio de una utilidad llamada `rpcgen`, la cual toma el archivo de definición de funciones RPCL del servidor y crea un conjunto de archivos con funciones de C, los cuales crean una capa de software que facilita las llamadas a funciones y la conversión de datos a XDR.

## 11.2. Librería VXI

Es una compilación de código fuente que permite establecer comunicación con instrumentos habilitados para ethernet y que usen el protocolo VXI11, para Linux. Permite la comunicación con una amplia gama de instrumentos como osciloscopios, analizadores lógicos generadores de funciones. Incluye además dos utilidades interactivas para enviar y recibir comandos SCPI a los instrumentos.

El autor del código, Steve D. Sharples, logró conseguir los archivos RPCL originados de la especificación del protocolo VXI-11.

```
program DEVICE_CORE {
    version DEVICE_CORE_VERSION {
        Create_LinkResp create_link (Create_LinkParms) = 10;
        Device_WriteResp device_write (Device_WriteParms) = 11;
        Device_ReadResp device_read (Device_ReadParms) = 12;
        Device_Error destroy_link (Device_Link) = 23;
    } = 1;
} = 0x0607AF;
```

<b>Función API</b>	<b>Descripción</b>
<code>create_link</code>	Establece un enlace a un dispositivo lógico dentro de un servidor VXI-11
<code>device_write</code>	Envía un comando (típicamente SCPI) a un instrumento
<code>device_read</code>	Lee datos de un instrumento
<code>destroy_link</code>	Libera el enlace establecido por <code>create_link</code> y libera los recursos utilizados.

Cuadro 1: Funciones VXI-11 básicas

### 11.2.1. Archivos componentes

La librería consiste en los siguientes archivos de código fuente:

`vxi11.x` el cual es una fusión de los archivos RPCL `vxi11core.rpcl` y `vxi11intr.rpcl`. Es una base ligera para `vxi11` sobre `rpc`. Si se ejecuta `rpcgen` en este archivo, se generan los archivos de C y cabeceras, a partir de los cuales se pueden escribir programas de C para comunicación con instrumentos ethernet. Al ejecutar la utilidad `rpcgen` sobre este archivo, se genera los archivos de C con sus respectivas cabeceras los cuales contienen las funciones necesarias (ver tabla 1) para establecer comunicación con instrumentos mediante el protocolo RPC.

`vxi11_user.cc` y `vxi11_user.h` incluye varias funciones, entre ellas 4 funciones clave para facilitar la programación al usuario: `vxi11_open()`, `vxi11_close()`, `vxi11_send()` and `vxi11_receive()`. Incluyen funciones que encapsulan la complejidad de las llamadas nativas RPC.

`vxi11_cmd.c` código fuente para utilidad interactiva de línea de comando llamada `vxi11_cmd` que permite enviar comandos y consultas a un instrumento `vxi11`, el cual se localiza por medio de su dirección IP.

`Makefile` archivo con guio de instrucciones par la utilidad `make`, que permite construir el programa utilidad `vxi11_cmd`. Simplemente se teclea `make` para compilar el código. Luego `make clean` para eliminar antiguos archivos con código objeto (.o) y ejecutables. Por último con `make install` se copia la utilidad `vxi11_cmd` a `/usr/local/bin`.

### 11.2.2. Utilidades

Dentro del código fuente se encuentra el archivo de nombre `Makefile`. Este archivo sirve de entrada para el comando `make` de linux, el cual permite generar la utilidad `vxi11_cmd`, de la siguiente manera,

```
make clean
make install
```

La primera instrucción permite remover antiguos archivos de código objeto de construcciones previas. El último comando permite construir la utilidad `vxi11_cmd` y copiarla a la ruta `/usr/local/bin`.

### 11.3. Construcción de la librería libvxi11.so

El código fuente incluye el archivo `makefile` que le permite al comando `make` generar un ejecutable `vxi11_cmd`, una utilidad de depuración que sirve para enviar y recibir comandos textuales (`scpi`) a un instrumento.

Si se desea emplear el código fuente en una aplicación que requiere la comunicación con instrumentos LAN, el programador o desarrollador tiene dos opciones.

1. Integrar el código fuente a la aplicación.
2. Generar una librería a partir del código fuente y enlazar la librería con la aplicación.

La opción 1 es la más simple siempre y cuando se desarrolle una aplicación en lenguaje C/C++. Solo requiere incluir dentro del código fuente de la aplicación a desarrollar los archivos `vxi11.h`, `vxi11_user.h` y `vxi11_user.cc`, que forman parte del código fuente para VXI-11. El programador podrá emplear las funciones básicas de la que se muestran en la tabla 1. La aplicación se debe compilar como un todo.

La opción 2 permite generar una librería que contenga las funciones de comunicación y a la cual la aplicación puede cargar al momento de iniciar la ejecución. Presenta la ventaja que se compila y construye el código fuente VXI11 sólo una vez, cuando se construye la librería. Además es posible llamar a las funciones de esta librería escrita en C/C++ desde otros lenguajes de programación, como Java, C# (CSharp) o Python.

En linux, se pueden generar dos tipos de librería: estática o dinámica. Una librería estática es un archivo con extensión `.a`, esta se incluye como parte del programa durante la *compilación*. Es el equivalente a un archivo `.lib` en Windows.

Una *librería dinámica de objetos compartidos (shared objects)* es un archivo con extensión `.so` se construye aparte de la aplicación principal y se carga cuando arranca la aplicación.

Para generar la librería `libvxi11.so` se empleará el archivo de código fuente `vxi11_user.cc` el cual incluye las funciones de usuario necesarias para establecer comunicación VXI11 con los instrumentos Lan. Este archivo requiere los archivos de cabecera fuente `vxi11_user.h` y `vxi11.h`.

El archivo `vxi11_user.cc` es un archivo de C++, ya que presenta funciones sobrecargadas, esto es, funciones con el mismo nombre pero con distintos numero y tipo de parámetros. La librería debe generarse con el compilador de C++ de linux, `g++`.

El compilador de C++ genera una especie de transformación en los nombres de las funciones que se conoce como *name mangling*. Simplemente el compilador le agrega un prefijo al nombre de la función que indica el numero y tipo de argumentos que recibe la función. Esto lo realiza para permitir funciones con el mismo nombre pero distintos argumentos.

Por ejemplo, el nombre de la función `int vxi11_open_device(const char *ip, CLINK *clink, char *device)` es codificado por el compilador `g++` como

```
_Z15vxi11_open_linkPKcPP6CLIENTPP15Create_LinkRespPc
```

## 12. Anexos

### 12.1. Script de Bash para carga automática de firmware

Listing 1: Listado programa

```
#include <stdlib.h>
#include <stdio.h>

int main()
{
    printf("Hola Gafo");

    return 0;
}
```