

## **rev 0.1 Abril 11 del 2006**

**Moisés Humberto Silva Salmerón** <moy@ivsol.net>

El presente documento se encuentra liberado bajo la licencia de documentación GNU (GNU FDL) en la versión mas reciente aplicable. Mas información en <http://www.gnu.org/licenses/fdl.txt>

TODO:

- gráficos para explicar el flujo de señalización
- gráficos para explicar como utilizar testcall y un cable cruzado E1 para probar internamente las tarjetas.

## **NOTAS SOBRE MFCR2 CON ASTERISK Y AVANTEL**

### **Consideraciones sobre este documento.**

El presente documento se encuentra basado en mi experiencia personal con la señalización MFC/R2 con Avantel. NO soy un experto en señalización y puede contener errores. La tabla de señalización está basada en un trazado de señalización que me fué enviado por Avantel cuando intentábamos detectar los errores que teníamos en el establecimiento de las llamadas. Se agradece cualquier contribución al presente documento.

### **¿Qué es MFCR2 y para qué sirve?**

Multi Frequency Compelled Region 2 signaling. Esto es, señalización región 2 dirigida por medio de frecuencias (tonos). La señalización R2 es de tipo CAS (Channel Associated Signaling) y fue desarrollada en los 60s, pero aún usada en Europa, Asia, Latino América y Australia. Se encuentra pseudo-definida por la ITU (International Telecommunications Union). Digo pseudo-definida debido a que cada país ha creado variaciones según sus necesidades, pero en términos generales opera de la misma forma. Definiéndolo concretamente, es señalización utilizada para el establecimiento de llamadas. Gráficamente sería:

Central Telefónica 1 <===== Señalización MFCR2 =====> Central Telefónica 2

Cabe mencionar que, sin contar las variaciones de cada país, existen 3 tipos de R2.

R2 Digital, generalmente utilizado en troncales E1 en sistemas PCM (Pulse Coded Modulation), y en México usamos ALAW como método de codificación.

R2 Análogo

R2 por pulsos, utilizado normalmente donde la transmisión es lenta, como conexiones satelitales.

### **¿Cómo Funciona?**

MFCR2 es señalización "peer to peer". Lo que significa que ambos lados son iguales desde el punto de vista de la señalización, es decir, no existe un "cliente" y un "servidor". Aquí ambos lados del enlace operan de la misma forma. R2 utiliza las facilidades digitales de las líneas E1 para llevar a cabo su cometido. Se encuentra fuera del alcance de este documento analizar lo que es una E1, sin embargo, podemos decir que es un medio de transmisión que divide su funcionalidad en 32 canales, usualmente referidos como TS0 - TS31. De estos 32 canales, se utiliza el canal 0 para información como el framing, sincronía, alarmas y cosas concretas para cada país. Los canales 1-15 y 17 a 31 son para voz y/o datos, y el canal 16 es para la señalización (Aquí viaja R2).

Para la realización de una llamada, R2 utiliza 2 tipos de señales:

Señales Supervisoras.

Se encargan de supervisar el estado de la línea o enlace. Son enviadas por medio de 4 bits que son comúnmente conocidos como bits ABCD.

Señales de inter-registro.

Se encargan de la inicialización y control de la llamada. A través de estas señales se envían los dígitos ANI y DNIS, como veremos mas adelante. Estas señales son tonales, y es donde se transmiten las señales MF (Multi Frequency).

Lo primero a entender son las señales supervisoras. Las señales supervisoras son sumamente simples. La siguiente tabla describe las señales "Forwards" y "Backwards". El término Forwards se refiere a la señalización vista desde quien llama hacia el destino, y Backwards, naturalmente lo contrario. La siguiente tabla corresponde a la señalización que puede apreciar ocurre entre una central Ericsson de Avantel, y la librería MF2R2 escrita por Steve Underwood.

Forwards ABCD	Nombre	Backwards ABCD	Nombre
1 0 0 1	Idle	1 0 0 1	Idle
1 0 0 0	Seizure	1 1 0 1	Seizure Acknowledgment
1 0 0 1	Clear Back	1 0 0 1	Clear Back
1 0 0 1	Clear Forward	1 0 0 1	Clear Forward
0 1 0 1	Answer	0 1 0 1	Answer
1 1 0 1	Blocking	1 1 0 1	Blocking

Seizure (También conocido como "toma de troncal")

Señal enviada para solicitar la iniciación de una nueva llamada.

Seizure Ack

Señal enviada para confirmar la recepción del Seizure, e indicarle al otro extremo que puede empezar la señalización por tonos (MF).

Answer

Indica que la llamada ha sido contestada.

Clear Backwards

Señal enviada para indicar que quien recibió la llamada ha colgado.

Clear Forwards

Señal enviada para indicar que el originador de la llamada ha colgado.

Blocking

Señal enviada para indicar que no se recibirán llamadas.

Las señales de inter-registro son utilizadas para establecer detalles mas concretos de la llamada, como el ANI y DNIS. El ANI (Automatic Number Identification) es mejor conocido como servicio de identificador de llamada, que no es más que la detección del número de quien nos llama (CallerID). El DNIS (Dialed Number Identification

Service) es el servicio proveído con el objetivo de informar cual fue la DID (Direct Inward Dial) que se ha solicitado, es decir, qué número se marcó para originar la llamada.

Hay 3 tipos de señales de inter-registro:

**R2-Compelled:** Significa que cuando un par de tonos es enviado desde el switch (Forward), la señal permanece hasta recibir una respuesta (ACK) por medio de otro par de tonos. Cuando el switch recibe esta respuesta, cancela la primer señal. El otro lado debe detectar este silencio y a su vez suprimir la señal que envió anteriormente como respuesta. El switch detectará este silencio y enviará la segunda señal, y así sucesivamente.

**R2-Non-Compelled:** Desconozco los detalles, pero parece ser que los tonos son enviados por pulsos, no permanecen de forma constante hasta recibir respuesta como en R2-Compelled.

**R2-Semi-Compelled:** Una mezcla de las anteriores. Las señales Forwards son enviadas en pares de tonos constantes, pero las señales Backwards son enviadas en forma de pulsos.

Cualquier referencia a R2 en este documento indica R2-Compelled.

Las señales de inter-registro son "in-band", lo que significa que las señales son enviadas en el mismo canal que la voz (en términos generales, en el mismo canal que los datos). Las señales de inter-registro forwards se dividen en grupo I y grupo II, mientras que las backwards en grupo A y grupo B. A continuación describo cada una de estas señales. Al final del documento encontraremos un trazado de una llamada concretada entre un PBX Asterisk con librería libunicall y la central Ericsson de Avantel.

Señales Forward	Señales Backward
<p><b>Grupo I</b>                      Son señales utilizadas para la transmisión de los dígitos ANI y DNIS.</p> <p>Al menos con avantel los dígitos son representados del 1 al 10. El 10 significa 0. El 15 significa fin de dígitos. (no comprobado con Avantel)</p>	<p><b>Grupo A</b>                      Indican recepción de dígitos y cambio de señalización.</p> <p>A-1 significa envíame el siguiente dígito                      A-3 Fin de recepción de dígitos (Address Complete) cambio a señales de grupo B                      A-4 Congestión                      A-5 Envío de categoría de quien llama                      A-6 Fin de recepción de dígitos</p>
<p><b>Grupo II</b>                      Utilizadas para la transmisión de la categoría de quien llama.</p> <p>I-1 Suscriptor sin prioridad                      I-2 a I-9 Suscriptor con prioridad                      I-11 a I-15 uso nacional, desconozco el uso en México.</p>	<p><b>Grupo B</b>                      Enviadas para confirmar la recepción de las señales de grupo II o para proveer información del destino. Estas señales se encuentran siempre precedidas por una señal A-3 (ver Grupo A)</p> <p>B-1 En avantel indica abrir path de audio                      B-3 Línea ocupada                      B-4 Congestión                      B-5 Número desconocido                      B-6 Línea libre de cargos</p>

## Como se integra MFC/R2 con Asterisk?

En el mejor de los casos. Integrar Asterisk con MFCR/2 es sencillo. Bastaría seguir el excelente artículo publicado por Ariel Molina (<http://zarzamora.com.mx/asterisk/17>) y todo debería funcionar de maravilla. Algunos no fuimos afortunados y nos topamos con un inconveniente; un error en la programación de la librería de C que interpreta el protocolo R2 (libmfcr2). Algo mas de información sobre este error, y como fué solucionado puede ser visto en <http://moy.ivsol.net/> , sin embargo, puede bastarles con saber que hace una semana Steve Underwood liberó una versión corregida del código.

Si se toman el tiempo para leer el artículo de Ariel, encontrarán que escribe un pequeño esquemático en donde muestra cómo se relacionan Asterisk, libunicall, libmfcr2 etc. El mismo que podemos encontrar en el sitio de Steve Underwood (autor del sistema Unicall). Sin embargo, aquí explicaré un poco más a detalle cómo trabajan en conjunto estos componentes para hacer funcionar una llamada. Un esquemático un poco mas completo sería:

[bajo nivel] [alto nivel]  
PSTN <----> zaptel card <----> zaptel driver <----> kernel linux <----> libmfcr2 <-----> libunicall <----> chan\_unicall <----> Asterisk

Los drivers de zaptel son "character devices" registrados en el kernel linux. El driver registra algunos IOCTLs que luego el kernel expone para que las aplicaciones en "User Space" puedan comunicarse con los drivers de zaptel. En este caso, libmfcr2 se comunica con el kernel a través de estos IOCTLs para ser informado del estado de los bits ABCD, recepción de información etc. en la tarjeta zaptel. Así mismo utiliza llamadas al sistema, como write(), para escribir a descriptores de archivo pertenecientes a los drivers de zaptel. Cuando hay cambio en los bits ABCD al estado "seizure" (toma de troncal), libmfcr2 inicia una nueva llamada, comunicándose con libunicall que es quien expone un API bien definido para la realización y aceptación de llamadas de forma independiente del protocolo de señalización (en este caso MFCR2). libunicall comunica a una aplicación de mas alto nivel (en este caso chan\_unicall, parte integral de Asterisk) los eventos de bajo nivel reportados por libmfcr2. libmfcr2 utiliza la librería de spandsp para responder con tonos e identificar correctamente los tonos MF (multifrequency) para R2, recordemos que MFCR2 es un protocolo de señalización dirigido por tonos. En versiones recientes de libmfcr2 se requiere también de libsupertone, una librería para la generación y detección de tonos de supervisión (tono de ocupado, desconexión etc). Estos tonos pueden ser generados también por Asterisk (indications.conf).

Prestando atención a la descripción de la cadena de comunicación detectamos dos puntos a mi parecer muy importantes:

1. libunicall es una excelente abstracción. En este caso le permite comunicarse a Asterisk por medio de una tarjeta zaptel utilizando señalización MFCR2 con otras redes, pero de igual forma es posible implementar el uso de cualquier otra tarjeta y/o protocolo de señalización sin que Asterisk lo note, puesto que Asterisk seguirá contento mientras se le siga exponiendo el mismo API por parte de libunicall (la belleza de las abstracciones). La afirmación anterior es parcialmente cierta, debido a que algunas partes de chan\_unicall.c contienen IOCTLs específicos para zaptel, por lo que estas pequeñas partes tendrían que ser reemplazadas por código para la nueva tarjeta a usar. Sin

embargo en cuanto a señalización (en este caso MFCR2), libunicall es una abstracción limpia y completa.

2. libunicall puede ser utilizado por cualquier otra aplicación, no solo Asterisk. Para muestra basta utilizar la mini aplicación testcall.c que viene con la distribución de libunicall. Es una aplicación que simplemente genera o acepta llamadas (igual que Asterisk a través de chan\_unicall, solo que Asterisk las acepta y las conecta con otros canales, además de proveer servicios mas complejos). Esta aplicación es muy útil para probar tu instalación y descartar problemas en chan\_unicall o Asterisk mismo. No sé si exista ya una implementación, pero sería posible por ejemplo, utilizar libunicall con YATE (yet another telephony engine, <http://yate.null.ro/pmwiki/>).

## Configuración y solución de problemas.

Haciendo referencia nuevamente al artículo de Ariel. Se menciona que utilizar testcall es una lata. Yo diría que si te funciona a la primera, puedes darte ese lujo. Si encuentras fallas, testcall es muy útil, porque te permite aislar el problema, ya que para utilizar testcall no necesitas de Asterisk. La versión mas reciente de unicall ya trae soporte para un archivo de configuración para testcall, y de esta manera no tienes que modificar los "defines" en el código fuente. Otra herramienta que es de utilidad para encontrar problemas es "zttool", incluida con los drivers de zaptel. Si no la tienes, dirígete a los fuentes de zaptel y teclea "make zttool". En caso de que no tengas testcall tampoco, ve a los fuentes de libunicall y teclea "make testcall". Ahora, ya con esas dos importantes herramientas a la mano, pasemos a hacer una descripción de la forma en la que debería estar configurado tu sistema para recibir llamadas por medio de una E1 usando CAS y MFCR2.

Este archivo es leído por los drivers de zaptel para modificar el comportamiento de la tarjeta. Para hacer que zaptel lea el archivo de configuración, ejecutamos el comando "ztcfg -vv" Las opciones son para que zaptel reporte con mensajes lo que encuentra y configura. A continuación muestro una configuración simple:

```
span=1,1,0,cas,hdb3
cas=1-10:1101
dchan=16
defaultzone=us
loadzone=us
```

Esta configuración es la que tengo para un servidor con una tarjeta digium de 4 puertos. En este caso solo utilizo 1 puerto. Zaptel llama a sus puertos "spans". Puedes revisar cuántos puertos son identificados haciendo:

```
ls /proc/zaptel/
```

Eso debe mostrarte el número de spans (puertos). Recuerda que si tienes compilado el modulo ztdummy de zaptel verás 1 span mas. Ahora, si tu haces algo como cat /proc/zaptel/1 veras datos importantes del span. Mas adelante los explicaremos. De momento veamos qué significa cada opción en zaptel.conf.

El parámetro "span" es para configurar los puertos. Los puertos necesitan ser configurados al menos con un una prioridad de "timing", un LBO, framing y coding; cada valor separado por una coma. Así que la primer línea del archivo anterior se lee: "configura el span 1 como fuente de timing con prioridad 1, LBO 1, framing CAS y coding HDB3".

**Timing:** Si recibes el timing desde el otro punto a donde este conectado el puerto, es recomendable utilizar un valor de 1 aquí. Indicando que utilizarás el puerto como fuente primaria de timing.

**LBO:** Line Build Out, es un entero que debe ser seleccionado de la siguiente tabla.

0: 0 db (CSU) / 0-133 feet (DSX-1)

1: 133-266 feet (DSX-1)

2: 266-399 feet (DSX-1)

3: 399-533 feet (DSX-1)

4: 533-655 feet (DSX-1)

5: -7.5db (CSU)

6: -15db (CSU)

7: -22.5db (CSU)

hasta donde sé debe estar muy relacionado con la longitud del enlace. Normalmente usamos cero.

**Framing:** CAS, Channel Associated Signaling. En wikipedia podrás encontrar a que se refiere. CAS es el framing en donde viajará MFCR2.

**Coding:** HDB3 (High Density Bipolar-3 Zeros). Es la forma en la que la serie de unos y ceros serán transmitidos físicamente. Se encuentra altamente relacionado con el medio de transmisión a utilizar.

El parámetro cas=1-10:1101 significa que los canales del 1 al 10 utilizarán señalización CAS, y los bits ABCD serán inicializados al valor 1101 respectivamente. La línea "dchan=16" indica que el canal 16 será quien lleve la señalización. Los valores defaultzone y loadzone son utilizados para los tonos. Sin embargo desconozco el impacto que pueden tener en una configuración

Puedes encontrar información mas detallada en el archivo de ejemplo incluido con zaptel. Los puertos son muy versátiles, incluso puedes conectarte a Internet mediante el uso de HDLC (High level Data Link Control).

Una vez propiamente configurado. Asegurate de ejecutar "ztcfg -vv". Esto provocará que el archivo de configuración sea leído. Ahora un "cat /proc/zaptel/x" donde x es el número de span que configuraste, deberá mostrarte todos los canales que pertenecen a ese SPAN. Considerando la configuración anterior, los canales del 1 al 10 deberían mostrar algo como:

Span 1:TE4/0/1 "T4XXP (PCI) Card 0 Span 1" HDB3/ClockSource

1 TE4/0/1/1 CAS

.....

10 TE4/0/1/2 CAS

....

10 TE4/0/1/16 HDLCFS

TE4 es el modelo de la tarjeta, los números separados por diagonales son el número de tarjeta, span y canal respectivamente.

Ahora, ejecutando el programa "zttool", debe mostrarte cada uno de los SPANs de tu computadora. Indicando si están configurados o no. Lo primero que debes ver es que los spans conectados deben mostrar el estado "OK". Es decir, ninguna alarma. Selecciona el span anteriormente configurado y debe mostrarte una pantalla con los bits TxA, TxB, TxC, TxD y RxA, RxB, RxC, RxD. Que son el estado de los bits ABCD transmitidos y recibidos. Si configuraste bien el span, los bits Tx deben estar en 1101 (Blocked) para los canales del 1 al 10, o en 1001 (Idle) si tienes iniciado Asterisk. Si estás conectado correctamente a Telmex, Avantel o alguna otra compañía telefónica, y ya te han habilitado el servicio de telefonía por E1, los bits Rx deben estar en 1001 (Idle) para tantos canales (líneas) como hayas contratado. Si no te aparecen bits ABCD significa que configuraste incorrectamente la tarjeta.

La configuración de Asterisk es muy simple, solo copia el archivo unicall.conf que viene con la distribución de Unicall a la carpeta /etc/asterisk, o donde sea que tengas tus archivos de configuración. Los parámetros mas importantes son:

```
protocolclass=mfcr2
protocolvariant=mx,0,4,7
channel=1-10
```

El primer parámetro procolclass indica que usaremos señalización MFCR2. Para esto debes tener instalada la librería libmfcr2.c, unicall buscará el .so (shared object) en /usr/lib/unicall/protocols/protocol\_mfcr2.so para utilizarlo para interpretar y generar la señalización adecuada. El path anterior puede variar dependiendo de la manera en la que compilaste libunicall y libmfcr2. El path anterior es válido si fue configurado con --prefix=/usr. El segundo parámetro protocolvariant es para especificar la variante de México para el protocolo mfcr2. Las opciones están separadas por comas.

mx: indica variante para México

0: indica 0 dígitos ANI para recibir. Debes consultar con tu compañía si te enviará o no, y cuantos, dígitos ANI.

4: Cantidad de dígitos DNIS

7: Máscara de opciones

Respecto al último parámetro, las opciones soportadas son:

<b>Decimal</b>	<b>Binario</b>	<b>Opción</b>
1	00000001	Generar tono de marcado. Usualmente no necesario, debido a que la compañía de teléfonos suele proveerlo.
2	00000010	Tono de desconexión. ídem.
4	00000100	Ringback tone?
8	00001000	Obtener el ANI antes que el DNIS. El comportamiento normal es obtener primero el DNIS. Con Avantel se obtiene primero el DNIS
16	00010000	Aceptar llamadas inmediatamente, saltando el uso de señales B y Grupo II.

Cualquier combinación deseada de opciones puede ser aplicada simplemente realizando una operación "OR" lógica entre los números binarios y poniendo el número equivalente decimal después de la última coma en protocolvariant.

Teniendo zaptel.conf correctamente configurado podemos usar testcall para diagnosticar problemas con las llamadas. El archivo de configuración de testcall puede ser como el siguiente:

```
# file created by moy@ivsol.net
# this file is sensitive to blank lines, dont put them!
# comments are allowed if the first char in line is #
# circuits is the number of channels in zaptel span
# valid values for on-offered are: accept,answer,busy,unassigned,congested,oos,random
# caller parameter tells if the test is in caller or callee mode
destination-no 013331339767
protocol-class mfc2
protocol-variant mx,10,4
protocol-end cpe
caller no
originating-no 30025860
on-offered accept
circuits 1-10
```

**destination-no:** indica el número que será marcado en caso de que el parámetro "caller" sea igual a "yes"

**protocol-class:** indica la señalización a usar

**protocol-end:** ignórenlo, para MFCR2 da igual

**caller:** valores aceptados "yes" para probar en modo de llamada saliente, cualquier otro valor para modo de llamada entrante

**originating-no:** callerid que será enviado en caso de que "caller" sea igual a "yes"

**on-offered:** que hacer cuando una llamada sea recibida, en caso de que "caller" sea diferente de "yes"

**circuits:** que circuitos (canales) utilizar. Si configuraste zaptel.conf como lo indiqué anteriormente, el valor sería 1-10, indicando que serán utilizados los canales del 1 al 10. Te recomiendo que si vas a probar con llamadas salientes solo utilices un canal a la vez, es decir, algo como 1-1. Si no intentará generar muchas llamadas a la vez y es algo confuso seguirle la pista a todos los mensajes. Puedes especificar múltiples rangos simplemente repitiendo el parámetro. De tal forma que si tuvieras configurados los 31 canales, pudieras especificarlos (saltándote el 16, que es un canal de señalización.):

```
circuits 1-15
```

```
circuits 17-31
```

Supondremos que se ha configurado el programa con "caller" igual a "no". Ahora solo basta con correr el programa testcall para empezar con el "debugging". Verás un mensaje consistente, cada segundo, diciendo "Main Thread". Lo que indica que el programa está corriendo correctamente. En este momento libunicall ha abierto (con la llamada al sistema open(/dev/zap/channel)) todos los canales que tu le hayas indicado mediante el parámetro circuits. Y ahora se encuentra esperando que el otro punto del enlace (usualmente la compañía telefónica) le solicite iniciar una llamada mediante la toma de troncal, recordemos que la toma de troncal implica un cambio en los bits ABCD, por lo que el primer paso a comprobar ( si no estás recibiendo llamadas ) es verificar desde "zttool" si los Rx Bits están cambiando correctamente (revisando la tabla de señales supervisoras).

Ahora puedes marcar desde un teléfono convencional a una de las DIDs que se te hayan asignado, testcall debe mostrar muchos mensajes sobre las señales que envía y recibe. Recomiendo que edites el archivo testcall.c y busques una línea como esta:

```
logging_level= 2 & UC_LOG_SEVERITY_MASK
```

y la sustituyas por:

```
logging_level= UC_LOG_SEVERITY_MASK
```

Para que muestre tantos mensajes de debugging como sea posible.

Si quieres asegurarte si el problema se encuentra en la compañía telefónica o en tu configuración, y cuentas con al menos dos puertos en tu tarjeta zaptel. Intenta haciendo un cable cruzado para E1. Simplemente se conectan los cables 1 con 4 y 2 con 5. Aquí encontrarás esquemáticos bastante buenos, checa el que dice "E1/PRI crossover cable: RJ48C/RJ48C" <http://www.chebucto.ns.ca/Chebucto/Technical/Manuals/Max/max6000/gs/cables.htm>

Una vez hecho el crossover, conéctalo entre los dos puertos de la tarjeta. Si el cable está bien hecho, al menos deberías ver "OK" con zttool en ambos spans. Posteriormente modifica testcall.c para que acepte el archivo de configuración como parámetro, la última vez que vi estaba "hard coded" a usar testcall.conf. Una vez hecho esto crea un archivo de configuración con "caller yes" y otro con "caller no". Por último asegúrate que configures los

circuítos de forma que los canales del span 1 coincidan con los del span 2 (suponiendo que conectaste entre si los span 1 y 2). Esto es, si configuras el circuito 1-1 en SPAN 1, debes configurar al menos circuito 32-32 en el SPAN 2. Antes de ejecutar las 2 instancias de testcall, verifica con zttool que los bits ABCD se encuentren en "blocked".

Inicia el primer "testcall" usando el archivo de configuración en modo "caller no", revisa nuevamente con zttool el estado de los bits ABCD, y debes tener 1001 (Idle) en los circuítos configurados con testcall. Ahora, sin detener testcall, inicia otra instancia de testcall con el archivo en modo "caller yes" y la llamada debe efectuarse perfectamente.

**happy debugging!**