Comprobación del funcionamiento del sintetizador RacalDana 3101

D. Cordobés, L. Barbas,
J.A. López Pérez, C. Almendros

Informe Técnico IT - OAN 2007 - 05

CONTENIDO

I. Introducción

El Centro Astronómico de Yebes dispone de un sintetizador de frecuencias RacalDana 3101, que aunque tiene cierta antigüedad, goza de muy altas prestaciones. Durante el traslado de los equipos de VLBI desde el radiotelescopio de 14m al de 40m [1] se detectó un fallo al arrancar el sintetizador que nos impidió utilizarlo en ese momento. Ante la inminencia de los trabajos del traslado nos vimos obligados a posponer su reparación.

En este informe se detallan los problemas encontrados, soluciones propuestas y pruebas realizadas.

II. Problemas encontrados y soluciones propuestas

Al encender el sintetizador se generaba el código de error #02, el cual se corresponde con un error en el chequeo inicial de la ROM [2]. Lo primero que se hizo fue invocar la función especial #71, la cual hace un chequeo de la ROM. El resultado de esta función fue positivo, lo que significa que la ROM funciona correctamente. Sin embargo, tras invocar la función especial #76 se observó que los contenidos del banco #5 de memoria estaban corruptos, por lo que se hizo uso de la función especial #72 que carga en la memoria la configuración que el usuario desee (se eligió una frecuencia de salida de 200MHz y una potencia de salida de 0dBm). Un chequeo posterior de la función especial #76 dio resultados positivos.

Adicionalmente, se detectó que el teclado del sintetizador en ciertas ocasiones se bloquea. Esto se debe a la antigüedad del equipo, que provoca que algunos botones de su teclado numérico se queden presionados tras pulsarlos, haciéndose necesario desencajarlos manualmente de esa posición.

III. Pruebas realizadas

Se ha estudiado la precisión en frecuencia y en potencia del sintetizador en el laboratorio empleando tanto su referencia interna de frecuencia como una externa. Tras encender el sintetizador, se ajustó el estándar interno de frecuencia empleando el tornillo ajustable en su panel trasero bajo la etiqueta "*Internal Standard frecuency adjustment*". La forma de proceder se detalla en el apartado 7 del manual de mantenimiento del equipo [3] y se muestra en la **Figura 1**. Se trata de sincronizar la señal de 10MHz *ext ref out,* generada por el oscilador interno del sintetizador RacalDana, con otra señal de 10MHz. En este caso se ha empleado la señal de 10MHz generado por el distribuidor Quartzlock A5 que a su vez la obtiene del máser de Hidrógeno del CAY. Una vez montado el banco de medida, el estándar interno de frecuencia se ajusta girando el tornillo hasta que la sinusoide que aparece en la pantalla del osciloscopio deje de moverse horizontalmente. En ese momento, la potencia de la señal *ext ref out* del RacalDana ha de tener una nivel de 0dBm ± 2dB.

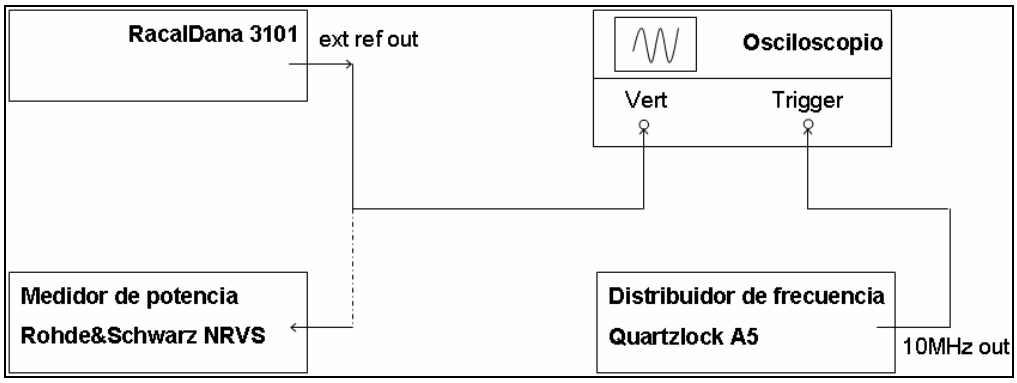

Figura 1. Banco de medida para ajustar la referencia interna del sintetizador RacalDana 3101

Ajustada de esta manera la referencia interna del sintetizador se hicieron medidas de frecuencia y potencia empleando tanto su referencia interna como una externa.

3.1) Medidas con la referencia interna de frecuencia

Se hace uso de un contador con el que se mide la frecuencia de salida del RacalDana y de un medidor de potencia, cuya interconexión se muestra en la **Figura 2**.

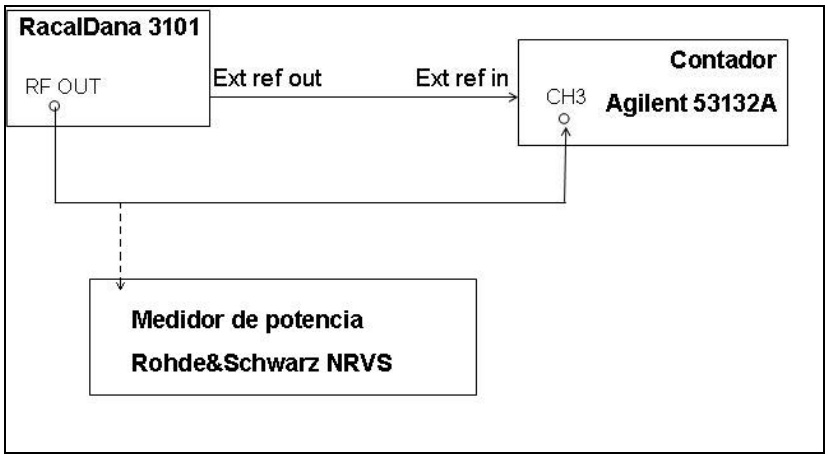

Figura 2. Banco empleado para medir la potencia y la frecuencia del sintetizador RacalDana 3101 utilizando su referencia interna

3.1.1) Medidas de la frecuencia de salida

Según el procedimento detallado por el fabricante en el manual de mantenimiento del RacalDana [3], el contador se configuró para que integrase 10 segundos y obtuviese la media. En la **Tabla 1** se muestra la diferencia entre la frecuencia comandada y la medida por el contador, así como la desviación en frecuencia. Los valores de frecuencia medidos están dentro del error de ±0.1Hz en la frecuencia de salida dado por el fabricante del RacalDana.

Frecuencia comandada [MHz] f_c	**Frecuencia medida [MHz] f_m**	**Desviación en frecuencia (ppm)** $\delta f = \frac{\lvert f_c - f_m \rvert}{f_c} \cdot 10^6$
200	199.999 999 99	$5e^{-5}$
400	399.999 999 99	$2.5e^{-5}$
600	599.999 999 97	$5e^{-5}$
800	799.999 999 98	$2.5e^{-5}$
1000	999.999 999 98	$2e^{-5}$
1200	1199.999 999 97	$1.7\ e^{-5}$
1300	1299.999 999 99	$8.3\ e^{-6}$

Tabla 1. Frecuencias medidas [MHz] haciendo uso de la referencia interna de frecuencia

3.1.2) Medidas de la potencia de salida

En la **Tabla 2** se muestra la diferencia entre la potencia comandada y la medida por el detector de potencia (en la fórmula de la desviación las potencias están en watios). Los valores de potencia medidos están dentro del margen de error de ±0.4dB para frecuencias menores de 650MHz y de ±0.7dB para frecuencias mayores de 650MHz.

Potencia comandada [dBm] P_c	**Potencia medida [dBm] P_m**	**Desviación en potencia (%)** $\delta P = \frac{\lvert P_c - P_m \rvert}{P_c} \cdot 100$
0	0.25	5.92
5	5.21	4.95
10	10.13	3.04
15	14.95	1.14
19	18.84	3.62

Tabla 2. Potencias medidas [dBm] haciendo uso de la referencia interna de frecuencia

3.2) Medidas con una referencia externa de frecuencia

En este caso se empleó el distribuidor de frecuencia Quartzlock A5 (que recibe la señal del máser) para sincronizar el RacalDana y el contador (**Figura 3**).

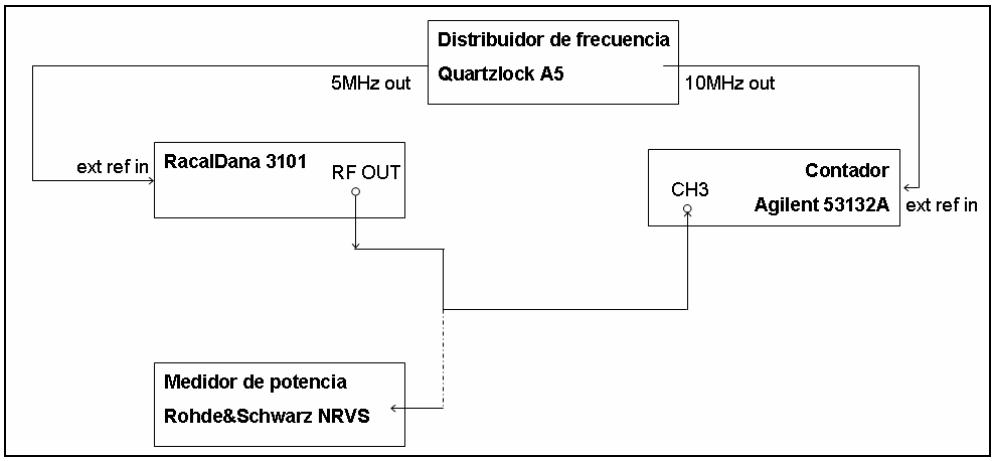

Figura 3. Banco empleado para medir la potencia y la frecuencia del sintetizador RacalDana 3101 utilizando una referencia externa

3.2.1) Medidas de la frecuencia de salida

Según el procedimento detallado por el fabricante en el manual de mantenimiento del RacalDana [3], el contador se configuró para que integrase 10 segundos y obtuviese la media. En la **Tabla 3** se muestra la diferencia entre la frecuencia comandada y la medida por el contador. Los valores de frecuencia medidos están dentro del error de ±0.1Hz en la frecuencia de salida dado por el fabricante del RacalDana.

Frecuencia comandada [MHz] f_c	**Frecuencia medida [MHz] f_m**	**Desviación en frecuencia (ppm)** $\delta f = \frac{\lvert f_c - f_m \rvert}{f_c} \cdot 10^6$
200	199.999 999 99	$5e^{-5}$
400	399.999 999 99	$2.5e^{-5}$
600	599.999 999 99	$1.7e^{-5}$
800	799.999 999 98	$2.5e^{-5}$
1000	999.999 999 98	$2e^{-5}$
1200	1199.999 999 99	$8.3\ e^{-6}$
1300	1299.999 999 99	$7.7\ e^{-6}$

Tabla 3. Frecuencias medidas [MHz] haciendo uso de una referencia externa de frecuencia

3.2.2) Medidas de la potencia de salida

En la **Tabla 4** se muestra la diferencia entre la potencia comandada y la medida por el detector de potencia (en la fórmula de la desviación las potencias están en watios). Los valores de potencia medidos están dentro del margen de error de ±0.4dB para frecuencias menores de 650MHz y de ±0.7dB para frecuencias mayores de 650MHz.

Potencia comandada [dBm] P_c	**Potencia medida [dBm] P_m**	**Desviación en potencia (%)** $\delta P = \frac{\lvert P_c - P_m \rvert}{P_c} \cdot 100$
0	0.26	6.17
5	5.2	4.71
10	10.13	3.04
15	14.91	2.05
19	18.88	2.73

Tabla 4. Potencias medidas [dBm] haciendo uso de una referencia externa de frecuencia

IV. Control remoto del equipo

El RacalDana 3101 se puede controlar y monitorizar remotamente a través del bus GPIB. Se han hecho pruebas análogas a las presentadas en el Apartado 3 de este informe que han dado resultados positivos de lo que se deduce que la comunicación GPIB del sintetizador funciona correctamente. En la **Figura 4** se muestra el banco de medida empleado.

El software para controlar y monitorizar el RacalDana 3101 ha sido desarrollado en C++ por Laura Barbas y se muestra en el Apéndice.

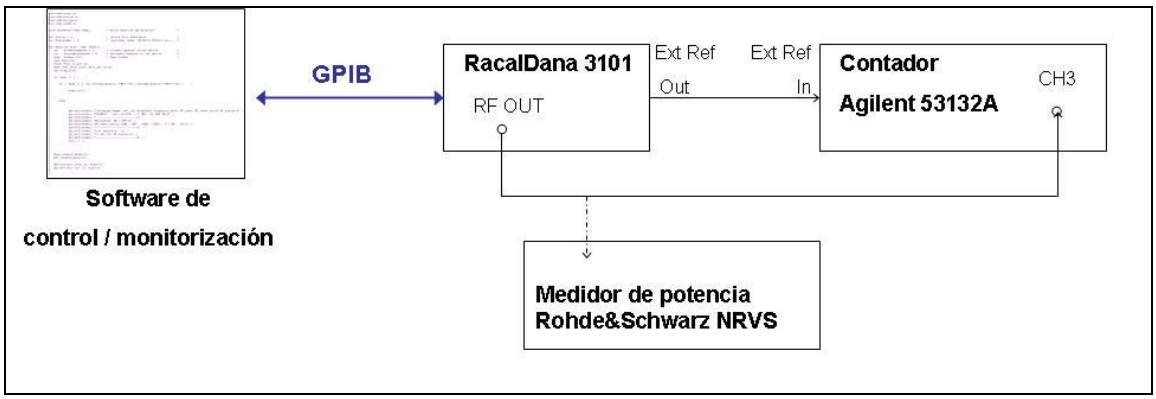

Figura 4. Banco empleado para medir remotamente la potencia y la frecuencia del sintetizador RacalDana 3101

V. Conclusiones

Tras lo expuesto anteriormente, se concluye que a fecha de la redacción de este informe (06/02/2007), el sintetizador Racal Dana 3101 funciona correctamente con la salvedad del teclado numérico, en el que hay que vigilar que ninguna tecla se quede pulsada.

VI. Bibliografía

[1] D. Cordobés, P. de Vicente, J. Fernández, C. Almendros, J.Mª Yagüe, *Traslado de los equipos de VLBI al radiotelescopio de 40m*, Informe Técnico IT- OAN 2007-1.

[2] *RACAL-DANA Operator's manual*, RACAL Electronics Group

[3] *RACAL-DANA Maintenance manual*, RACAL Electronics Group

Apéndice: Software para controlar/monitorizar el RacalDana 3101

1) rd3101.h

```
/**************************************************************************
 * ARIES21 - Antena Radiomilimétrica Espanola Siglo XXI
 *
 * Copyright (C) 2004
 * Observatorio Astronómico Nacional, Spain
 *
 * This library is free software; you can redistribute it and/or modify it under
 * the terms of the GNU Library General Public License as published by the Free
 * Software Foundation; either version 2 of the License, or (at your option) any
 * later version.
 *
 * This library is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY FITNESS
 * FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more
 * details.
 *
 * You should have received a copy of the GNU Library General Public License
 * along with this library; if not, write to the Free Software Foundation, Inc.,
 * 675 Massachusetts Ave, Cambridge, MA 02139, USA. Correspondence concerning
 * APEX should be addressed as follows:
 *
 * who          when        what
 * --------     ----------  ----------------------------------------------
 * lbarbas      2004/09/28      created
 */

#ifndef _RD3101_H_
#define _RD3101_H_

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <unistd.h>
#include <./c++/3.4/exception>
#include <ugpib.h>

using namespace std;

/**
 * @mainpage RD3101
 * This code implements the C++ component of the <i>Racal-Dana synthesizer Model 3101</i>
 * designed for the testing of communication equipment over the frequency range from 10
 * kHz to 1.3GHz.
 * <P>The component has been declared in a <b>header file</b> and has been implemented in a
 * <b>cpp file</b>.
 * <P>The operation has been made via the GPIB.</P>
 *
 * @author Laura Barbas l.barbas@oan.es
 * @date 2004-10-04
 */

/**
 * The component class <i>RD3101</i> contains the declaration for the constructor, the
destructor,
```

```
 * several methods to obtain or set the frequency or amplitude of the synthesizer and
finally private
 * members which hold any parameters of the synthesizer.
 *
 * <BR>This class defines its own exception.
 */
class RD3101
{
   public:
        /**
         * Defines an own exception of RD3101 class. Uses as base
        <code><b>exception</b></code> class.
         * RD3101Exception allows to obtain the error message, status and error code when an
        exceptional     * circumstance happens.
         */
        class RD3101Exception : public exception
        {
                public:
                        /**
                         * Constructor. Stores the error message, the variable
                        <var>ibsta</var> and
                         * the variable <var>iberr</var> in the corresponding attributes.
                         */
                        RD3101Exception(const char * ex, const int & status, const int &
                        err);

                        /**
                         * Gets the message related to the exceptional event.
                         * @return [char *] The message related to the exceptional event.
                         */
                        const char * ShowMsg() {return m_msgException;}

                        /**
                         * Gets <var>ibsta</var>: the GPIB variable that holds status.
                         * @return [int] ibsta.
                         */
                        int Status() {return m_ibstaException;}

                        /**
                         * Gets <var>iberr</var>: the GPIB variable that holds error code.
                         * @return [int] iberr.
                         */
                        int Err (){return m_iberrException;}

                private:
                        char m_msgException[100];/**< Stores a message related to error
                        cause.*/
                        int m_ibstaException;/**< Stores the GPIB variable
                        <var>ibsta</var>.*/
                        int m_iberrException;/**< Stores the GPIB variable
                        <var>iberr</var>.*/
        };

        /**
         * Constructor with default arguments:
         * <UL>
         * <LI>Sets the board and the device addresses</LI>
         * <LI>Clears the GPIB interface functions with IFC</LI>
         * <LI>Sets Remote Eneable line (board to listener)</LI>
         * <LI>Sets local mode. The device will return to remote mode
         *     the next time it is addressed by the controller in charge.</LI>
         * </UL>
         */
        RD3101(const int & boardID = 0, const Addr4882_t & devID = 19,
               const double & freqMAX = 1300, const double & freqMIN = 0.01,
                const double & ampMAX = 19, const double & ampMIN = 0) throw (RD3101Exception
                &);
        /**
         * Destructor: do-nothing function.
         */
        ~RD3101() throw (RD3101Exception &);
```

```
/**
 * @brief Sets the frequency in MHz.
 *
 * This function cheks if the frequency is in the valid range (from 10kHz
 * to 1.3GHz). In affirmative case makes up a command with the frequency value
 * and sends it to the device (Racal-Dana). Otherwise throws an exception.
 *
 * @param freq [<i>double</i>]   Frequency: must be in MHz.
 */
void SetFrequency(const double & freq) throw (RD3101Exception &);

/**
 * @brief Sets the amplitude in dBm.
 *
 * This function cheks if the amplitude is in the valid range (from 0 dBm
 * to 19 dBm). In affirmative case makes up a command with the amplitude value
 * and sends it to the device (Racal-Dana). Otherwise throws an exception.
 *
 * @param amp [<i>double</i>]   Amplitude: must be in dBm.
 */
void SetAmplitude(const double & amp) throw (RD3101Exception &);

/**
 * @brief Gets output frequency in MHz.
 *
* This function calls <code>Tokens()</code> function. Later takes out the
frequency value of
 * <var>m_tokFQ</var> attribute.
 * @return [double] frequency value.
 * @see Tokens()
 */
double Frequency() throw (RD3101Exception &);

/**
 * @brief Gets output amplitude in dBm.
 *
 * This function calls <code>Tokens()</code> function. Later takes out the amplitude
value of
 * <var>m_tokAP</var> attribute.
 * @return [double] amplitude value.
 * @see Tokens()
 */
double Amplitude() throw (RD3101Exception &);

/**
 * @brief Gets de instrument-data data string of 163 ASCII characters.
 *
 * Sends to device the addressed command <code>ID</code> in order to receive the
data string.
 * The string determines the current instrument settings and is stored in
<var>m_instData</var>.
 * @return [char *] instrument-data data string
 */
char *InstData() throw (RD3101Exception &);

/**
 * @brief Gets de instrument-status data string of 27 bytes.
 *
 * Sends to device the addressed command <code>IS</code> in order to receive the
data string.
 * The string determines the error code numbers and the status byte. It is stored in
 * <var>m_instStatus</var>.
 * @return [char *] instrument-status data string
 */
char *InstStatus() throw (RD3101Exception &);

/**
 * @brief Initializes the instrument settings.
 *
 * Sends to device the addressed command <code>IP</code> in order to initialize the
```

```
         * instrument settings.
         */
        void Initialize() throw (RD3101Exception &);

    private:
        Addr4882_t m_devID;   /**< Stores device address.*/
        int m_boardID;        /**< Stores board address.*/
        double m_freqMAX;     /**< Stores maximum frequency input allowed in MHz. Default
        value: 1300*/
        double m_ampMAX;      /**< Stores maximum amplitude input allowed in dBm. Default
        value: 19*/
        double m_freqMIN;     /**< Stores minimum frequency input allowed in MHz. Default
        value: 0.01*/
        double m_ampMIN;      /**< Stores minimum amplitude input allowed in dBM. Default
        value: 0*/
        char m_instData[192]; /**< Stores the instrument-data data string.*/
        char m_instStatus[32];/**< Stores the instrument-status data string.*/

        //The instrumen-data data string tokens
        char m_tokUnitID[5];  /**< Stores unit ID code.*/
        char m_tokConfString[5];/**< Stores configuration string.*/
        char m_tokStatString[5];/**< Stores status string.*/
        char m_tokActFunct[9];/**< Stores active function string.*/
        char m_tokFQ[17];     /**< Stores output frequency in MHz.*/
        char m_tokFS[17];     /**< Stores frequency step size in MHz.*/
        char m_tokAP[11];     /**< Stores amplitude in current display units.*/
        char m_tokAS[10];     /**< Stores amplitude step size, either dB or volts units.*/
        char m_tokGPIBEnt[5]; /**< Stores GPIB entry mode string.*/
        char m_tokContrBit[9];/**< Stores secondary control bit string.*/
        char m_tokSpecFunct[3];       /**< Stores active special function code string.*/
        char m_tokFA[17];     /**< Stores sweep start fequency in MHz.*/
        char m_tokFB[17];     /**< Stores sweep stop frequency in MHz.*/
        char m_tokSoftPack[14];       /**< Stores software package number.*/
        char m_tokOoL[9];     /**< Stores out of lock line status string.*/
        char m_tokPowSup[9];  /**< Stores power supply monitor status string.*/

        /**
         * @brief Gets the tokens of the instrument-data data string.
         *
         * This function calls <code>InstData()</code> function. Later takes out all tokens
        of
         * <var>m_instData</var> attribute and stores it in the corresponding attributes.
         *
         * @see InstData()
         */
        void Tokens() throw (RD3101Exception &);

        /**
         * Assignment operator: do-nothing function.
         * @param constant reference to a RD3101 object.
         * @return reference to a RD3101 object.
         */
        RD3101 & operator=(const RD3101 &);

        /**
         * Copy constructor: do-nothing function.
         * @param constant reference to a class object.
         */
        RD3101(const RD3101 &);

};

#endif  //_RD3101_H_
```

2) rd3101.cpp

```
/*************************************************************************
 * ARIES21 - Antena Radiomilimétrica Espanola Siglo XXI
 *
 * Copyright (C) 2004
 * Observatorio Astronómico Nacional, Spain
 *
 * This library is free software; you can redistribute it and/or modify it under
 * the terms of the GNU Library General Public License as published by the Free
 * Software Foundation; either version 2 of the License, or (at your option) any
 * later version.
 *
 * This library is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY FITNESS
 * FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more
 * details.
 *
 * You should have received a copy of the GNU Library General Public License
 * along with this library; if not, write to the Free Software Foundation, Inc.,
 * 675 Massachusetts Ave, Cambridge, MA 02139, USA. Correspondence concerning
 * APEX should be addressed as follows:
 *
 * who          when        what
 * --------     ----------  ----------------------------------------------
 * lbarbas     2004/09/28     created
 */

#include "rd3101.h"

RD3101::RD3101(const int & boardID, const Addr4882_t & devID,
               const double & freqMAX, const double & freqMIN,
                const double & ampMAX, const double & ampMIN) throw (RD3101Exception &)
{
        m_boardID = boardID;
        m_devID = devID;
        m_freqMAX = freqMAX;
        m_freqMIN = freqMIN;
        m_ampMAX = ampMAX;
        m_ampMIN = ampMIN;

        // Clear the GPIB interface functions with IFC
        SendIFC(m_boardID);
        if (ibsta & ERR)
                throw RD3101Exception("[RD3101::RD3101] Error SendIFC", ibsta, ERR);

        // Set Remote Eneable line (board to listener)
        if(ibsre(m_boardID,1) & ERR)
                throw RD3101Exception("[RD3101::RD3101] Error Enable Remote", ibsta, ERR);

        // The board to go to local mode. If the system controller is asserting the REN
        line,
        // then devices on the bus will return to remote mode the next time they are
        addressed
        // by the controller in charge.
        if(ibloc(m_boardID) & ERR)
                throw RD3101Exception("[RD3101::RD3101] Error Enable Local", ibsta, ERR);
}

RD3101::~RD3101() throw (RD3101Exception &){}

void RD3101::SetFrequency(const double & freq) throw (RD3101Exception &)
{
        // Change the frequency
        // Range: 10kHz - 1300 MHz
        if(freq > m_freqMAX)
                throw RD3101Exception("[RD3101::SetFrequency] Frequency entered greater than
                1.3 GHz"
```

```
                                        , ibsta, ERR);
        if(freq < m_freqMIN)
                throw RD3101Exception("[RD3101::SetFrequency] Frequency entered less than 10
                kHz", ibsta, ERR);
        char cmd[16];
        sprintf(cmd,"FQ%fMZ",freq);
         //Addresses the device (m_devID) as listener, then writes data (FQ...) onto the bus
        Send(m_boardID,m_devID,cmd,strlen(cmd),NLend);
        if(ibsta & ERR)
                throw RD3101Exception("[RD3101::RD3101] Error Send Frequency", ibsta, ERR);
}

void RD3101::SetAmplitude(const double & amp) throw (RD3101Exception &)
{
        // Change the amplitude
        // Range: +19 dBm - 0 dBm (2V - 224 mV)
        if(amp > m_ampMAX)
                throw RD3101Exception("[RD3101::SetAmplitude] Amplitude input exceeds +19
                dBm", ibsta, ERR);
        if(amp < m_ampMIN)
                throw RD3101Exception("[RD3101::SetAmplitude] Amplitude input less than 0
                dBm", ibsta, ERR);
        char cmd[16];
        sprintf(cmd,"AP%fDB",amp);
         //Addresses the device (m_devID) as listener, then writes data (AP...) onto the bus
        Send(m_boardID,m_devID,cmd,strlen(cmd),NLend);
        if(ibsta & ERR)
                throw RD3101Exception("[RD3101::RD3101] Error Send Amplitude", ibsta, ERR);
}

double RD3101::Frequency()throw (RD3101Exception &)
{
        //Get frequency
        double freq = 0;

        char * tok;
        const char delim[]="Q";

        Tokens(); // private function
        tok = strtok(m_tokFQ, delim);
        tok = strtok(NULL,delim);
        freq = atof(tok);

        return freq;
}

double RD3101::Amplitude() throw (RD3101Exception &)
{
        //Get amplitude
        double amp = 0;

        char * tok;
        const char delim[]="P";

        Tokens(); //private function
        tok = strtok(m_tokAP, delim);
        tok = strtok(NULL,delim);
        amp = atof(tok);

        return amp;
}

char *RD3101::InstData() throw (RD3101Exception &)
{

        // Get the instrument-data data string

         //Addresses the device (m_devID) as listener, then writes data (ID) onto the bus
         Send(m_boardID,m_devID,(void *)"ID",2L,NLend);
        if(ibsta & ERR)
                throw RD3101Exception("[RD3101::RD3101] Error Send ID", ibsta, ERR);
```

```
        //Performs the necessary addressing, then reads data from the device specified by
        m_devID
        // and stores in m_instData
        Receive(m_boardID,m_devID,m_instData,163,STOPend);
         if(ibsta & ERR)
                throw RD3101Exception("[RD3101::RD3101] Error Receive ID", ibsta, ERR);
        m_instData[ibcnt]='\0';

        return m_instData;
}

char *RD3101::InstStatus() throw (RD3101Exception &)
{

        // Get the instrument-status data string

        //Addresses the device (m_devID) as listener, then writes data (IS) onto the bus
         Send(m_boardID,m_devID,(void *)"IS",2L,NLend);
        if(ibsta & ERR)
                throw RD3101Exception("[RD3101::RD3101] Error Send IS", ibsta, ERR);

        //Performs the necessary addressing, then reads data from the device specified by
        m_devID
        // and stores in m_instStatus
        Receive(m_boardID,m_devID,m_instStatus,27,STOPend);
         if(ibsta & ERR)
                throw RD3101Exception("[RD3101::RD3101] Error Receive IS", ibsta, ERR);
        m_instStatus[ibcnt]='\0';

        return m_instStatus;
}

void RD3101::Initialize() throw (RD3101Exception &)
{
        //Initializes the instrument

        //Addresses the device (m_devID) as listener, then writes data (IP) onto the bus
         Send(m_boardID,m_devID,(void *)"IP",2L,NLend);
        if(ibsta & ERR)
                throw RD3101Exception("[RD3101::RD3101] Error Send IP", ibsta, ERR);

}

void RD3101::Tokens() throw (RD3101Exception &)
{
        char * token;
        const char delim[]=",";

        try {
                InstData();
        }
        catch(RD3101Exception &ex) {
                printf("%s\n",ex.ShowMsg());
        }

        //Gets tokens of the instrument-data data string (m_instData)
        strcpy(m_tokUnitID,strtok(m_instData, delim));
         strcpy(m_tokConfString,strtok(NULL, delim));
         strcpy(m_tokStatString,strtok(NULL, delim));
        strcpy(m_tokActFunct,strtok(NULL, delim));
         //Frequency
        strcpy(m_tokFQ,strtok(NULL, delim));
        strcpy(m_tokFS,strtok(NULL, delim));
        //Amplitude
        strcpy(m_tokAP,strtok(NULL, delim));
        strcpy(m_tokAS,strtok(NULL, delim));
        strcpy(m_tokGPIBEnt,strtok(NULL, delim));
        strcpy(m_tokContrBit,strtok(NULL, delim));
        strcpy(m_tokSpecFunct,strtok(NULL, delim));
        strcpy(m_tokFA,strtok(NULL, delim));
```

```
        strcpy(m_tokFB,strtok(NULL, delim));
       strcpy(m_tokSoftPack,strtok(NULL, delim));
       strcpy(m_tokOoL,strtok(NULL, delim));
       strcpy(m_tokPowSup,strtok(NULL, delim));

}

RD3101::RD3101Exception::RD3101Exception(const char * ex, const int & status, const int &
err): exception()
{
       strncpy(m_msgException,ex,100);
       m_ibstaException = status;
       m_iberrException = err;
}
```

3) prog.cpp

```
/* ARIES21 - Antena Radiomilimétrica Espanola Siglo XXI
 *
 * Copyright (C) 2004
 * Observatorio Astronómico Nacional, Spain
 *
 * This library is free software; you can redistribute it and/or modify it under
 * the terms of the GNU Library General Public License as published by the Free
 * Software Foundation; either version 2 of the License, or (at your option) any
 * later version.
 *
 * This library is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY FITNESS
 * FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more
 * details.
 *
 * You should have received a copy of the GNU Library General Public License
 * along with this library; if not, write to the Free Software Foundation, Inc.,
 * 675 Massachusetts Ave, Cambridge, MA 02139, USA. Correspondence concerning
 * APEX should be addressed as follows:
 *
 * who          when        what
 * --------     ----------  ----------------------------------------------
 * lbarbas      2004/09/28      created
 */

#include "rd3101.h"
#define MAX 256

int main(int argc,char **argv) {

       try {

              RD3101 dev1(0, 19);
              double frequency = 1300;
              double amplitude= 18;

              //frequency=atof(argv[1]);
              //amplitude=atof(argv[2]);
//            char cmd[MAX]="";
//            long len;
               char *resultData;
              char *resultStatus;

              printf("Starting prog...\n\n");
              //printf("Introduce comando:\t");
              //scanf("%s",cmd);
              //len = strlen(cmd);
              dev1.Initialize();
               printf("Initializing the instrument...\n\n");
              printf("Frequency of RD3101:%f\n", dev1.Frequency());
              printf("Amplitude of RD3101:%f\n", dev1.Amplitude());
```

```
                resultData = dev1.InstData();
                resultStatus = dev1.InstStatus();
                printf("Instrument-Data RD3101: %s\n",resultData);
                printf("Instrument-Status RD3101: %s\n",resultStatus);
                 printf("\nSetting frequency and amplitude...\n\n");
                dev1.SetFrequency(frequency);
                dev1.SetAmplitude(amplitude);
                printf("Frequency of RD3101:%f\n", dev1.Frequency());
                printf("Amplitude of RD3101:%f\n", dev1.Amplitude());
                resultData = dev1.InstData();
                resultStatus = dev1.InstStatus();
                printf("Instrument-Data RD3101: %s\n",resultData);
                printf("Instrument-Status RD3101: %s\n",resultStatus);

        }
        catch(RD3101::RD3101Exception &excep) {
                printf("%s\n",excep.ShowMsg());
                printf("ibsta:%d\n",excep.Status());
                printf("iberr:%d\n",excep.Err());
                printf("\n");
        }

        return 0;

}
```