Monitorización del máser de hidrógeno del CAY

P. de Vicente, A. Garrigues

Informe Técnico IT-OAN 2005-5

Índice

1. Introducción **2**

2. Montaje de medida **2**

3. Configuración del contador y recogida de datos **3**

4. Base de datos del Maser **12**

5. Envío de los datos diarios a la EVN **20**

6. Evolución del máser KVARZ CH1-75 instalado en Yebes entre los años 1998 a 2004 **24**
6.1. Variación del flujo de hidrógeno molecular y detector de fase con el tiempo . . 24
6.2. Variación de la frecuencia del máser con el tiempo 24

7. Estudio del mejor tiempo de integración para obtener el error relativo en frecuencia **28**

8. Estudio de la estabilidad de fase del sistema GPS - máser **32**

1. Introducción

En el CAY disponemos de un máser de hidrógeno CH1-75 que se emplea como referencia de tiempos y frecuencia. Para monitorizar y corregir el máser es necesario compararlo con la señal de un receptor GPS TrueTime XL-DC, tal y como se describe en los informes [8] y [9].

La monitorización se hace comparando 1 pulso por segundo (PPS) procedente del "Station Timing Unit" (STU), generado a partir de una señal de referencia procedente del máser y 1 PPS generado por el receptor GPS TrueTime XL-DC. Hasta ahora, la diferencia entre estos pulsos se medía con el contador interno del receptor GPS. La resolución de este contador, de 15ns, resultaba insuficiente ya que es inferior a las diferencias temporales que se producen entre ambos pulsos en medio día. La variación típica cada 10 minutos suele estar entre 1 y 5 ns. Para poder estudiar estas oscilaciones se ha sustituido el contador interno del GPS por un contador HP 53131A con una resolución de 0,25ns.

2. Montaje de medida

La figura 1 muestra el montaje de medida para comparar 1 PPS procedente del máser y otro procedente del receptor GPS. El contador HP 53131A dispone de 2 canales de entrada. Uno de los modos de operación permite medir el intervalo de tiempo transcurrido entre la señal del canal 1 y la del canal 2.

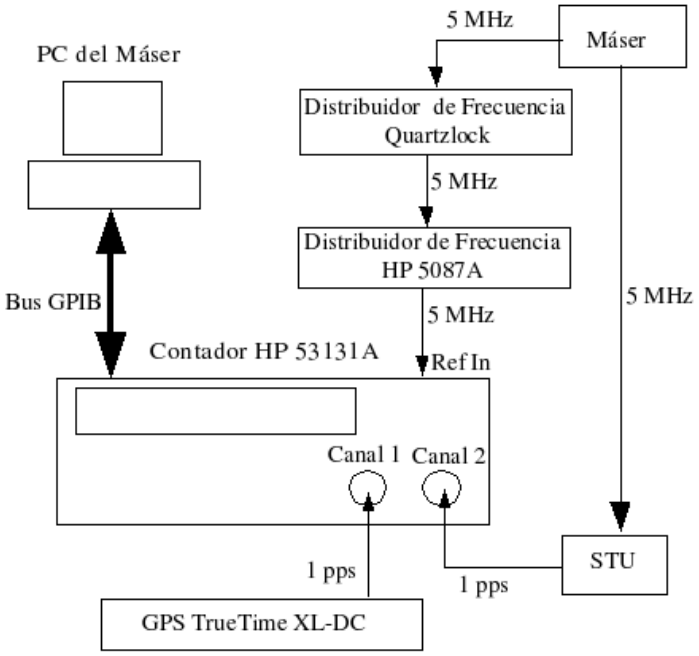

Figura 1: *Montaje de medida con el contador HP 53131A y el receptor GPS TrueTime XL- DC.*

El canal 1 se alimenta con una señal de 1 PPS procedente de una salida del panel trasero del receptor GPS y el canal 2 se alimenta con un PPS procedente del STU. Al STU a su vez, le llega

una señal de 5MHz procedente del máser. El contador utiliza una señal de referencia de 5 MHz procedente de un distribuidor de frecuencias HP 5087A. Este distribuidor de frecuencias genera las señales de salida a partir de una señal de 5MHz procedente del distribuidor de frecuencias "Quartzlock", que tiene como entrada una señal de 5MHz procedente del máser (Ver esquema de la figura 1)

La figura 2 muestra una imagen de los 2 pulsos, procedentes del GPS y de la STU, obtenida con un osciloscopio digital. La escala horizontal es 1μs/div y la vertical 1V/div. La señal superior corresponde al pulso del receptor GPS. Su amplitud es $\sim$2,8V y, aunque no se aprecia en la imagen, su anchura es $\sim 20\mu$s. La señal inferior proviene del STU, tiene una amplitud de $\sim 2,5$ V y una anchura de $\sim 1\mu$s.

3. Configuración del contador y recogida de datos

El contador HP 53131A dispone de un BUS GPIB que permite su control remoto. Para realizar las medidas automáticamente, se conectó un ordenador con una tarjeta PCI-GPIB de National Instruments. El ordenador corre en Linux (distribución Debian). La instalación y configuración del controlador se hizo siguiendo las instrucciones de National Instruments que se describen en: `http://www.ni.com/linux/ni488dl.htm`. Esta instalación solo es posible, en la fecha del informe, con núcleos Linux de la serie 2.4 o inferior. En dicho ordenador, en la cuenta "maser" se ha centralizado todo el software de medida.

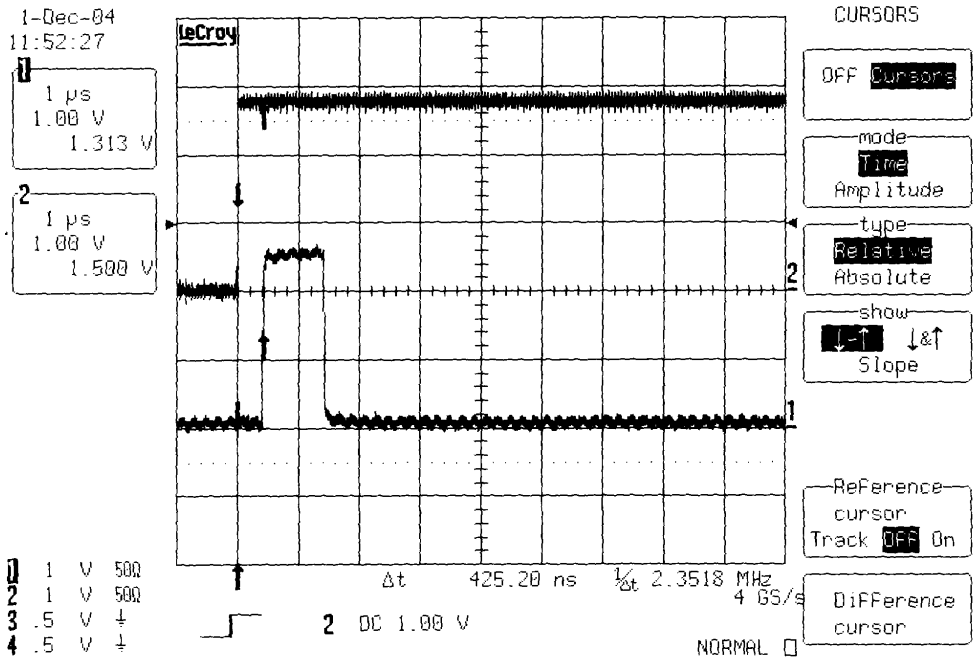

Figura 2: *Pulso generado por el GPS (arriba) y por la "Station Timing Unit" (abajo)*

Para automatizar las medidas se escribió un demonio que se encarga de leer la diferencia

temporal entre los pulsos (offset) medida por el contador cada segundo. Un demonio en Linux es un proceso que se pone en marcha al encender el ordenador y que se ejecuta permanentemente. De este modo nos aseguramos de que las medidas se pongan en marcha automáticamente incluso si el ordenador es apagado y luego reinicializado.

El demonio se ha escrito utilizando el metódo estándar de Debian y una plantilla ya disponible. Ello quiere decir que no es necesario seguir el procedimiento habitual que crea un proceso padre que luego se desdobla en un hijo empleando un fork. El código de inicio del demonio se muestra a continuación. Corresponde al archivo `/etc/init.d/counterdaemon`.

```
#! /bin/sh
#
# Author:    Ruben Bolaño, Amparo Garrigues
#
# Version:      @(#)skeleton  0.1  02-Nov-2003
#

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/home/maser/
     counter/bin
DAEMON=/home/maser/counter/bin/GpsMaserDif
NAME=GpsMaserDIf
DESC="Counter daemon"

# Gracefully exit if the package has been removed.
test -x $DAEMON || exit 0

# Read config file if it is present.
#[ -r /etc/default/$NAME ] && . /etc/default/$NAME

set -e

case "$1" in
  start)
        echo -n "Starting $DESC: $NAME"
        start-stop-daemon --start --quiet --make-pidfile --pidfile /var/run/$NAME.pid \
                --background --exec $DAEMON
        echo "."
        ;;
  stop)
        echo -n "Stopping $DESC: $NAME "
        start-stop-daemon --stop --quiet --pidfile /var/run/$NAME.pid \
        --exec $DAEMON
        echo "."
        ;;
        #reload)
        #
        #       If the daemon can reload its config files on the fly
        #       for example by sending it SIGHUP, do it here.
        #
        #       If the daemon responds to changes in its config file
        #       directly anyway, make this a do-nothing entry.
        #
        # echo -n "Reloading $DESC configuration..."
```

```
        # start-stop-daemon --stop --signal 1 --quiet --pidfile \
        #       /var/run/$NAME.pid --exec $DAEMON
        # echo "done."
  #;;
  restart|force-reload)
        #
        #       If the "reload" option is implemented, move the "force-reload"
        #       option to the "reload" entry above. If not, "force-reload" is
        #       just the same as "restart".
        #
        echo -n "Restarting $DESC: $NAME"
        start-stop-daemon --stop --quiet --pidfile \
              /var/run/$NAME.pid --exec $DAEMON
        sleep 1
        start-stop-daemon --start --quiet --pidfile \
              /var/run/$NAME.pid --exec $DAEMON
        echo "."
        ;;
  *)
esac
```

Este demonio, accede a la memoria compartida, inicializa el contador, lee constantemente del puerto GPIB los datos procedentes de aquel y los escribe en memoria compartida y en una base de datos MySQL. La escritura de datos en memoria compartida se hace para que estos estén disponibles para el equipo de VLBI a través de un programa servidor que escucha en un puerto TCP. Este servidor entrega los valores que hay en memoria compartida al cliente que se lo solicite.

A continuación se describe por orden el proceso que se lleva a cabo para la realización de las medidas.

- Durante el arranque del PC se reserva un área de memoria compartida con los datos de la comparación del maser con el GPS. Este área sólo contiene una estructura con cuatro variables, día juliano modificado, diferencia de tiempo entre pulsos obtenida cada segundo, media de la diferencia de tiempos en 10m y error cuadrático medio de los datos de 10m.

  La memoria compartida se reserva empleando la función `shmget()` y una clave IPC de SystemV que se obtiene a su vez a partir de una ruta y un identificador unívoco, empleando la función `ftok()`. Si fuera necesario reservar más de un área de memoria diferente, se deberían emplear claves diferentes y por tanto rutas e identificadores diferentes. Finalmente se “pega” el segmento de memoria compartida identificada por `shmid` al segmento de datos del proceso que llama a la función. La aplicación que se incluye a continuación muestra el proceso:

  ```
  // shared_defs.h
  #define COUNT_SIZE   4096  /* use multiples of a page (4096) */
  #define PATH "/tmp"
  #define ID (int)'c'
  ```

```
struct maserdata {
                double modifiedJulianDay;
                double gps_maser_diff;
                double gps_maser_diff_10m;
                double gps_rms_10m;
};

struct maserdata * shm_maserdata;

#include <shared_defs.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>

int main() {
        int size, shmid;
        key_t key;

        key = ftok(PATH, ID);
        size = COUNT_SIZE;

        struct shmid_ds str_shmid;  /* shared memory id struct */
        struct shmid_ds * buf;      /* shared memory id struct pointer */

        buf = & str_shmid;  /* make buf point to str_shmid */

        /* create, new key, permit all */
        shmid = shmget(key, size, (IPC_CREAT|0666));
        if (shmid == -1) {
                printf("get_mem: error allocating segment\n");
                exit(-1);
        }

        /* do a status on the shared memory segment */
        if (-1 == shmctl(shmid, IPC_STAT, buf)) {
                printf("get_mem: error checking size\n");
                exit(-1);
        }
        printf ("get_mem: id=%d, size is %d bytes\n",shmid,buf->shm_segsz);

        shmid = shmget (key, 0, 0 );
        if ( shmid == -1 ) {
                printf("get_mem: translating key failed\n");
                exit( -1);
        }

        shm_maserdata = NULL;
        shm_maserdata = (struct maserdata *) shmat ( shmid,
(char *) shm_maserdata, 0 );
        if ( ((char *)(-1))  == (char *) shm_maserdata ) {
                printf("get_mem: attaching memory segment failed\n");
```

```
                exit( -1);
        }
        return 1;
}
```

- La primera operación que realiza el demonio `GpsMaserDif` es obtener acceso a la memoria compartida.Dicho acceso se hace del modo habitual, y de un modo casi idéntico a como se crea.Observese que es necesario crear la clave IPC y emplear la función `shmget()` para asignar un segmento de memoria compartida. En este caso como el segmento ya está creado, sólo se comprueban los permisos de acceso. Finalmente la función `shmat()` "pega" el segmento de memoria compartida identificada por `shmid` al segmento de datos del proceso.

```
int shm_id;
key_t key;
int i;
key = ftok(PATH, ID);

shm_id = shmget(key,0,0);
if (shm_id == -1) {
printf("translating key failed\n");
exit(-1);
}
shm_maserdata = NULL;
shm_maserdata = (struct maserdata *) shmat ( shm_id, (char *) shm_maserdata, 0 );
if ( ((char *)(-1))  == (char *) shm_maserdata ) {
printf("attaching memory segment failed\n");
exit(-1);
}
```

- Se inicializa el contador y se entra en un bucle infinito que lee continuamente los datos del contador. Estas dos funciones se realizan a través de la clase `HPCounter`.

```
HPcounter counter14m = HPcounter(3);
counter14m.setModeComparator(1.3,1.3);

while(1) {
try {
                 dif = counter14m.readComparatorValue();
}
catch (HPcounter::HPcounterException & s){
if (DEBUG)
printf("Excepcion: %s, Ibsta: %d, Iberr: %d\n", s.ShowMsg(), s.ShowStat(),
s.ShowErr());
}
if (DEBUG)
printf("El valor es: %15.13f\n", dif);
try {
data_process(dif);
```

```
}
catch (int i)
{
if (DEBUG)
printf("Error de escritura en la bd");
continue;
}
}
```

- La clase `HPCounter` contiene las siguientes funciones:
  - El constructor de la clase que inicializa el contador, y establece la dirección GPIB en la que se encuentra. Los comandos de inicialización son los siguientes:

    ```
    writeToHP("*RST");
    writeToHP("*CLS");
    writeToHP("*SRE 0");
    writeToHP("*ESE 0");
    writeToHP(":STAT:PRES");
    ```

  - `writeToHP(char *message)`. Esta función escribe en el contador una cadena de caracteres a través del puerto GPIB. Si se produce un error se genera una excepción.
  - `readFromHP(char *data)`. Esta función lee una cadena del contador a través del puerto GPIB. Si se produce un error se genera una excepción.
  - `void setModeComparator(const double lev1, const double lev2)`. Esta función configura el contador para que mida el intervalo de tiempo transcurrido entre los dos pulsos. Los parámetros *lev1* y *lev2* indican el nivel a partir del cual se dispara el comienzo de la medida y el final de la medida. En esta función se configuran los canales para una impedancia de 50Ω, que los disparos se hagan con una rampa de subida y se use acoplamiento de corriente continua. La configuración para uno de los canales se hace del siguiente modo (la del otro canal es igual):

    ```
    writeToHP(":CONF:TINT");
    writeToHP("FUNC 'TINT'");
    // We switch off the levels
    writeToHP(":EVEN:LEV:AUTO OFF");
    // Signal in channel 1
    // Manually set the levels of trigger.
    sprintf(instr, ":EVEN:LEV %f V", lev1);
    writeToHP(instr);
    // Positive slope for the trigger
    writeToHP(":EVEN:SLOP POS");
    // Impedance of the signal: 50 Ohms
    writeToHP(":INP:IMP 50");
    writeToHP(":INP:COUP DC");
    writeToHP(":INP:ATT 1");
    writeToHP(":INP:FILT OFF");
    writeToHP(":EVEN:HYST:REL 0");
    ```

  - `double readComparatorValue()`. Esta función envía dos instrucciones solicitando la última lectura del comparador de tiempos, lee el valor y lo devuelve. Si se produce un error se genera una excepción.

- Los valores leídos se procesan del siguiente modo:

  - Los valores obtenidos cada segundo se almacenan junto con el día juliano modificado en memoria compartida. La unidad del intervalo de tiempo es el segundo.
  - Los valores se van acumulando en un fichero auxiliar. La alternativa consiste en definir un array donde se almacenan los datos.
  - Al cabo de 10 minutos se abre el archivo auxiliar se realiza la media y se obtiene el error cuadrático medio. Estos dos valores se almacenan en memoria compartida. Finalmente el valor medio cada 10 minutos, junto con el error cuadrático medio y el día juliano modificado (correspondiente al final del intervalo de los 10 minutos) se escriben en la base de datos.
  - El error cuadrático medio (rms) se calcula según la siguiente expresión:

$$s = \sqrt{\frac{(x_i - \overline{x})^2}{N-1}} \tag{1}$$

    donde x_i es el intervalo de tiempo transcurrido entre dos pulsos y medido por el contador cada segundo, $\overline{x}$ la media de dicho valor en 10 minutos y N el número de medidas realizadas.

- Por último, una función llamada `escribe_db` del demonio `GpsMaserDif` escribe el día juliano modificado, el valor leído del intervalo de tiempo entre los pulsos y el rms en la base de datos. Su código es el siguiente:

```
void escribe_db() {

        char query_str[512];
        char aux[100];
        MYSQL mysql;
        mysql_init(&mysql);
        if (!mysql_real_connect(&mysql,MYSQL_HOST,MYSQL_USERID,MYSQL_PASSWD,
MYSQL_DB,0,NULL,CLIENT_SSL))
        {
                if (DEBUG){
                        printf("Error: %s\n",mysql_error(&mysql));
                }
                throw 1;
        }
        strcpy(query_str,"INSERT INTO ");
        strcat(query_str,MYSQL_TABLE);
        strcat(query_str," VALUES(NULL,NULL,");
        sprintf(aux,"%10.4f",shm_maserdata->modifiedJulianDay);
        strcat(query_str,aux);
        strcat(query_str,",");
```

```
            sprintf(aux,"%15.12f",shm_maserdata->gps_maser_diff_10m);
            strcat(query_str,aux);
            strcat(query_str,",");
            sprintf(aux,"%18.15f",shm_maserdata->gps_rms_10m);
            strcat(query_str,aux);
            strcat(query_str,")");
            mysql_query(&mysql,query_str);
            mysql_close(&mysql);
    }
```

Otro programa denominado “del_countermem”, se encarga de borrar el espacio de memoria compartida previamente reservado. Su código es el siguiente:

```
#include <shared_defs.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>

int main() {
        int size, shmid;
        key_t key;

        key = ftok(PATH, ID);

        shmid = shmget (key, 0, 0 );
        if ( shmid == -1 ) {
                printf("del_mem: translating key failed\n");
                exit( -1);
        }
        if (shmctl(shmid, IPC_RMID, (struct shmid_ds *) 0) < 0)
        {
                printf("del_mem: can’t remove shared memory\n");
                exit(-1);
        }
        return 1;
}
```

Como la reserva de la memoria compartida y el demonio de medida del contador son dos procesos que se deben iniciar, por ese orden con el arranque del ordenador se deben situar en el directorio `/etc/init.d/`. Debian establece el modo en que se deben realizar los enlaces para que los guiones de cada una de estas aplicaciones `/etc/init.d/countermem` y `/etc/init.d/counterdaemon` se pongan en marcha. Las instrucciones,

```
update-rc.d countermem start 88 S .
update-rc.d counterdaemon start 89 S . stop 89 0 6 .
```

crean los enlaces correspondientes en los directorios `/etc/rc`, `/etc/rc0` y `/etc/rc6`. Los valores 88 y 89 de dichos comandos indican el orden en el que se inician durante el arranque. En este caso primero se incializa la memoria (88) y después se pone en marcha el demonio

de medida (89). Estos procesos son los últimos en ejecutarse en el arranque y los primeros en detenerse al apagar el ordenador.

Un servidor escrito en C++ denominado `GpsMaserSock` escucha en el puerto UDP 6544 y entrega la diferencia temporal entre los pulsos almacenada en la memoria compartida a cualquier cliente que la solicite en ese puerto. Su código se muestra a continuación:

```
#include <udpsocketserver.h>
#include <shared_defs.h>
#include <sys/shm.h>
#include <cstring>

using namespace std;

#define SERVER_UDP_PORT 6544
#define MAXMESG 2048
#define MAXBUF 14
#define DEBUG 0

void ini_sock();
int get_offset(char *off);

int main(){
int shm_id;
key_t key;
int i;
key = ftok(PATH, ID);
        shm_id = shmget(key,0,0);
if (shm_id == -1) {
printf("translating key failed\n");
exit(-1);
}
shm_maserdata = NULL;
shm_maserdata = (struct maserdata *) shmat ( shm_id, (char *) shm_maserdata, 0 );
if ( ((char *)(-1))  == (char *) shm_maserdata ) {
printf("attaching memory segment failed\n");
exit(-1);
}
ini_sock();
}

void ini_sock(){
int resp_len;
char mesg[MAXMESG];
char resp[MAXBUF];
Udpsocketserver udpSockSer = Udpsocketserver(SERVER_UDP_PORT);
udpSockSer.Bind();
for(;;){
udpSockSer.Read(mesg, MAXMESG, 0);
resp_len = get_offset(resp);
if (DEBUG)
printf("el offset es: %s\n", resp);
udpSockSer.Write(resp, resp_len, 0);
```

```
}
udpSockSer.Close();
}

int get_offset(char *off){
int n;
double offset;
offset = shm_maserdata->gps_maser_diff;
n=sprintf(off,"%15.13f", offset);
return n;
}
```

4. Base de datos del Maser

Hasta la fecha el valor de la media del offset durante 10 minutos, el rms y el día juliano modificado se almacenaban en ficheros mensuales que se guardaban a su vez en carpetas con su número de año correspondiente. Este sistema de almacenamiento no resultaba muy cómodo para el tratamiento posterior de los datos y por ello se optó por utilizar la base de datos MYSQL, al igual que en otros casos de monitorización de variables. De este modo, el almacenamiento y la recuperación de la información se hace de un modo más óptimo, es decir, con menor consumo de espacio y mayor velocidad. La información referente a la monitorización del máser se guarda en la base de datos *maserdb* que contiene 2 tablas:

- La tabla *maserdata* almacena la misma información que los antiguos ficheros (media del intervalo de tiempo cada 10 minutos, mjd y rms). Además se han añadido dos columnas para facilitar el tratamiento de los datos: el identificador que corresponde al número de entrada en la tabla y una marca temporal de 14 dígitos (año, mes, día, hora, minutos, segundos) que equivale al día juliano modificado. El cuadro 1 resume la denominación de cada campo y una descripción de su contenido.

- La tabla *maserstatus* guarda el estado del maser con los datos medidos cada mes. El cuadro 2 resume la denominación de cada campo y una descripción de su contenido. Esta información se registraba antiguamente en el cuaderno del maser, y en un fichero de texto.

Parámetro	Tipo de variable	Comentario
id	int(10) unsigned	Número de entrada en la tabla
ts	timestamp(14)	Marca temporal de 14 dígitos (año, mes, día, h, m, s)
mjd	double(10,4)	Día Juliano Modificado
maser_dps	double(10,4)	Media en 10m del intervalo de tiempo entre los pulsos (s)
rms	double(18,15)	Error cuadrático medio en 10m

Cuadro 1: *Parámetros de monitorización que se almacenan en la tabla* maserdata *de la base de datos* maserdb

Parámetro	Tipo de variable	Comentario
ts	timestamp(14)	Marca temporal de 14 dígitos (año, mes, día, hora, minuto, segundo)
mjd	double(10,4)	Día Juliano Modificado
24V	int(11)	Tensión de la alimentación de contínua de 24 V
18V	int(11)	Tensión de la alimentación de contínua de 18 V
0,5mA V	int(11)	Tensión 1 de la bomba iónica
0,5mA I	int(11)	Intensidad 1 de la boma iónica
50μA V	int(11)	Tensión 2 de la bomba iónica
50μA I	int(11)	Intensidad 2 de la boma iónica
5μA V	int(11)	Tensión 3 de la bomba iónica
5μA I	int(11)	Intensidad 3 de la boma iónica
12,6V	int(11)	Tensión de la alimentación de continua 12,6 V
-5V	int(11)	Tensión de la alimentación de continua -5 V
I purifier	int(11)	Intensidad de corriente del purificador
27V	int(11)	Tensión de la alimentación de continua 27 V
I HFOV	int(11)	Intensidad del disociador
IF level	int(11)	Nivel de señal de la frecuencia intermedia
Ph Det	int(11)	Fase de la señal de 5 MHz, respecto a la de resonancia
DAC	int(11)	indicador de la tensión del varactor
Synthes	int(11)	Frecuencia del sintetizador de frecuencia
H supply	float	Nivel del flujo de hidrógeno molecular
Temp Abm	float	Temperatura en la sala del máser
BAT1	float	Tensión en la batería 1
BAT2	float	Tensión en la batería 2
Comment	varchar(255)	Comentario

Cuadro 2: *Parámetros de monitorización que se almacenan en la tabla* maserstatus *de la base de datos* maserdb

Dado que los datos se guardan ahora en la base de datos, la extracción de los datos y la generación de las gráficas se realiza utilizando un programa escrito en Python a tal efecto. La extracción de datos y la generación de gráficas se hace utilizando el interfaz gráfico basado en Qt que se muestra en la figura 3. Este interfaz permite la selección de la fecha de comienzo y final de los datos y la elección de *día juliano modificado* o *fecha* para la representación en el eje de abscisas. Se pueden representar uno o dos parámetros en el eje de ordenadas. Para cada uno de los ellos se pueden introducir los valores máximo y mínimo a representar y el tipo de representación: *puntos* o *puntos unidos por líneas*. Se dan al usuario las siguientes opciones:

- Dibujar y no guardar nada.
- Dibujar y guardar la gráfica.
- Dibujar y guardar la gráfica y los datos.
- Extraer y guardar los datos.

En caso de querer guardar los datos, la gráfica o ambas cosas, se da la opción al usuario de elegir la ruta de almacenamiento. En el cuadro de texto de la parte inferior se muestran al usuario las acciones realizadas por el programa.

El código del programa en python se muestra a continuación:

```
#!/usr/bin/env python

from  plotOptions import *
import sys, _mysql
from qt import *
from string import *
import Gnuplot, Gnuplot.funcutils
from mx.DateTime import *
from threading import *
import os

class gplotX(Thread):
        def __init__(self):
Thread.__init__(self)
self.g0 = Gnuplot.Gnuplot()

        def run(self):
self.g0.reset()

class plotDb(plotOptions):
        def __init__(self):
plotOptions.__init__(self)
self.fechaIni.setDate(QDate.currentDate().addDays( -30 ))
self.fechaFin.setDate(QDate.currentDate())
self.fechaFin.setMaxValue(QDate.currentDate())

def slotReadData(self):
self.debuger.clear()
```

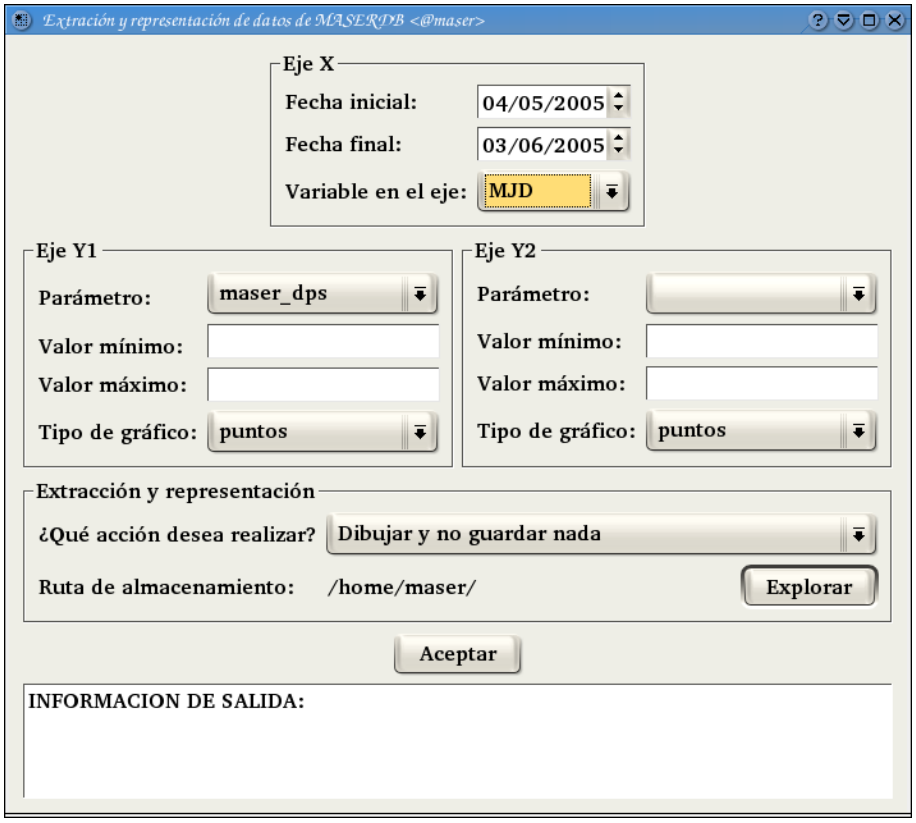

Figura 3: *Interfaz gráfico basado en QT empleado para la extracción y representación de datos de la base de datos* maserdb

```
self.debuger.insertItem('INFORMACION DE SALIDA:')
self.startDate = self.fechaIni.date()
self.finalDate = self.fechaFin.date()
self.xaxis = self.comboBoxOpX.currentText()
self.miny1 = self.lineEditMinY1.text()
self.maxy1 = self.lineEditMaxY1.text()
self.miny2 = self.lineEditMinY2.text()
                self.maxy2 = self.lineEditMaxY2.text()
self.y1axis = self.comboBoxEjeY1.currentText()
self.y2axis = self.comboBoxEjeY2.currentText()
self.y1PlotOp = self.comboBoxOpY1.currentText()
self.y2PlotOp = self.comboBoxOpY2.currentText()
self.accion = self.comboBoxAccion.currentText()
self.ruta = self.textLabelRuta2.text()
self.eligeDB()

def slotBuscaRuta(self):
self.ventanaRuta = QFileDialog("/home/maser/")
self.ventanaRuta.setMode(self.ventanaRuta.Directory)
self.ventanaRuta.connect(self.ventanaRuta, SIGNAL("dirEntered(const QString &)"),
self.escribeRuta)
self.ventanaRuta.show()

def escribeRuta(self):
self.ruta = str(self.ventanaRuta.dirPath()) + '/'
self.textLabelRuta2.setText(self.ruta)

def eligeDB(self):
yearStart = self.startDate.year()
monthStart = self.startDate.month()
dayStart = self.startDate.day()
yearFin = self.finalDate.year()
monthFin = self.finalDate.month()
dayFin = self.finalDate.day()

dateStart = DateTime(yearStart, monthStart, dayStart)
tsStart = dateStart.strftime('%Y%m%d' + '000000')
dateFin = DateTime(yearFin, monthFin, dayFin)
tsFin = dateFin.strftime('%Y%m%d' + '000000')
self.delta = dateFin.mjd - dateStart.mjd

self.dbCh = []
self.xaxisStr = str(self.xaxis)
if self.xaxisStr == 'fecha':
self.xaxisStr = 'ts'
self.y1axisStr = str(self.y1axis)
self.y2axisStr = str(self.y2axis)

if self.accion == 'Dibujar y guardar la grafica y los datos' or
    self.accion == 'Extraer y guardar los datos':
f1 = str(self.ruta) + 'dataXY.log'
f2 = str(self.ruta) + 'dataXY2.log'
else:
```

```
f1 = '/home/maser/data/dataXY.log'
f2 = '/home/maser/data/dataXY2.log'

self.fileOut = []
dbCh = []
qStrIni = 'SELECT ' + self.xaxisStr + ','
if self.y2axisStr == '':
if (self.y1axisStr == 'maser_dps' or self.y1axisStr == 'rms'):
dbCh.append('maserdata')
else:
dbCh.append('maserstatus')
qStr = qStrIni + self.y1axisStr + ' FROM '+ dbCh[0] + ' where ts >= ' + tsStart +
' && ts < ' + tsFin
self.fileOut.append(f1)
self.param = 2
self.extrae(qStr, self.fileOut[0])
else:
if (self.y1axisStr == 'maser_dps' or self.y1axisStr == 'rms') and
(self.y2axisStr == 'maser_dps' or self.y2axisStr == 'rms'):
self.dbCh.append('maserdata')
elif (self.y1axisStr != 'maser_dps' and self.y1axisStr != 'rms') and
(self.y2axisStr != 'maser_dps' and self.y2axisStr != 'rms'):
self.dbCh.append('maserstatus')
elif (self.y1axisStr == 'maser_dps' or self.y1axisStr == 'rms') and
(self.y2axisStr != 'maser_dps' and self.y2axisStr != 'rms'):
self.dbCh.append('maserdata')
self.dbCh.append('maserstatus')
else:
self.dbCh.append('maserstatus')
self.dbCh.append('maserdata')

if len(self.dbCh) == 1:
qStr = qStrIni +  self.y1axisStr + ', ' + self.y2axisStr +' FROM '+
self.dbCh[0] + ' where ts >= ' + tsStart + ' && ts < ' + tsFin
                      self.fileOut.append(f1)
self.param = 3
self.extrae(qStr, self.fileOut[0])
else:
qStr1 = qStrIni + self.y1axisStr + ' FROM '+ self.dbCh[0] + ' where ts >= '
 + tsStart + ' && ts < ' + tsFin
qStr2 = qStrIni + self.y2axisStr + ' FROM '+ self.dbCh[1] + ' where ts >= '
 + tsStart + ' && ts < ' + tsFin
self.fileOut.append(f1)
self.fileOut.append(f2)
self.param = 2
self.extrae( qStr1, self.fileOut[0])
self.extrae( qStr2, self.fileOut[1])

if self.accion[0:7] == 'Dibujar':
self.plot()

def plot(self):
self.gp = gplotX()
```

```
self.gp.start()
plotOpy1 = str(self.y1PlotOp)
plotOpy2 = str(self.y2PlotOp)
miny1Str = str(self.miny1)
maxy1Str = str(self.maxy1)
miny2Str = str(self.miny2)
maxy2Str = str(self.maxy2)
if plotOpy1 == 'puntos':
plotOpy1 = 'points'
else:
plotOpy1 = 'linespoints'
if plotOpy2 == 'puntos':
plotOpy2 = 'points'
else:
        plotOpy2 = 'linespoints'
if self.xaxisStr == 'ts':
self.gp.g0('set xdata time')
self.gp.g0('set timefmt "%Y%m%d%H%M%S"')
if self.delta > 180:
self.gp.g0('set format x "%m-%Y"')
else:
self.gp.g0('set format x "%d-%m-%Y"')
self.gp.g0('set grid x y')
if miny1Str != '' and maxy1Str != '':
self.gp.g0('set yrange [%s:%s]' %(miny1Str,maxy1Str))
self.gp.g0('set ytics nomirror')
if miny2Str != '' and maxy2Str != '':
self.gp.g0('set y2range [%s:%s]' %(miny2Str,maxy2Str))
self.gp.g0('set y2tics')
if len(self.fileOut) == 1:
if self.param == 3:
self.gp.g0.plot(Gnuplot.File(self.fileOut[0], using=(1,2),
with=plotOpy1, title= self.y1axisStr),
Gnuplot.File(self.fileOut[0], using=(1,3),with=plotOpy2,
axes='x1y2', title= self.y2axisStr))
else:
self.gp.g0.plot(Gnuplot.File(self.fileOut[0], using=(1,2),
with=plotOpy1, title=self.y1axisStr))
else:
self.gp.g0.plot(Gnuplot.File(self.fileOut[0], using=(1,2),
with=plotOpy1,title= self.y1axisStr),Gnuplot.File(self.fileOut[1],
using=(1,2),with=plotOpy2,axes='x1y2', title= self.y2axisStr))
if self.accion[10:17] == 'guardar':
fgraf = str(self.ruta) + 'graf.png'
self.gp.g0('set terminal png')
self.gp.g0('set output "'"%s"'"'  %fgraf)
self.debuger.insertItem('Grafica guardada en el fichero:')
self.debuger.insertItem(fgraf)
self.gp.g0.replot()
self.gp.g0('set output')
self.gp.g0('set terminal x11')
#self.gp.g0.replot()
#raw_input('Para continuar pulse Enter...\n')
```

```
def slotRmFiles(self):
if self.accion == 'Dibujar y guardar la grafica' or
self.accion == 'Dibujar y no guardar nada':
line =  os.popen('ls /home/maser/data/ | grep dataXY.log').readlines()
if line != []:
os.system('rm /home/maser/data/dataXY.log')
line =  os.popen('ls /home/maser/data/ | grep dataXY2.log').readlines()
if line != []:
os.system('rm /home/maser/data/dataXY2.log')

#print self.fileOut[0]
#if len(self.fileOut) == 1:
#os.system('rm %s' %self.fileOut[0])
#print 'Fichero de datos borrado'
#else:
# os.system('rm %s' %self.fileOut[0])
# os.system('rm %s' %self.fileOut[1])
# print 'Ficheros de datos borrados'

def extrae(self, query_str, fileO):
try:
                db = _mysql.connect(host = "localhost", user = "*******",
                passwd = "********", db = "******")
        except:
                print 'Error en la conexion a la base de datos'
                sys.exit(-1)
self.debuger.insertItem('Extraccion de los datos:')
self.debuger.insertItem(query_str)
db.query(query_str)
r = db.store_result()
filas = r.fetch_row(0)
db.close()
fOut = open(fileO,'w')
if len(filas) != 0:
if self.param == 2:
for f in filas:
if f[0]!= None and f[1]!= None:
x = atof(f[0])
y1 = atof(f[1])
fOut.write("%f %18.16f\n" %(x, y1))
elif self.param == 3:
for f in filas:
if f[0]!= None and f[1]!= None and f[2] != None:
        x = atof(f[0])
y1 = atof(f[1])
y2 = atof(f[2])
fOut.write("%f %18.16f %18.16f\n" %(x, y1, y2))

if self.accion == 'Dibujar y guardar la grafica y los datos' or
self.accion == 'Extraer y guardar los datos':
self.debuger.insertItem('Datos guardados en el fichero:')
self.debuger.insertItem(fileO)
```

```
else:
print 'No hay ninguna entrada en la base de datos para el intervalo
especificado o el intervalo especificado no es valido'
sys.exit(-1)
fOut.close()

def main(args):
ap = QApplication(sys.argv)
plotOp = plotDb()
ap.setMainWidget(plotOp)
plotOp.show()
ap.connect(plotOp.explorarButton,SIGNAL("clicked()"),plotOp.slotBuscaRuta)
ap.connect(plotOp.okButton,SIGNAL("clicked()"),plotOp.slotReadData)
ap.connect(ap, SIGNAL('lastWindowClosed()'), plotOp.slotRmFiles)
ap.exec_loop()

if __name__ == "__main__":
       main(sys.argv)
```

5. Envío de los datos diarios a la EVN

Con el fin de facilitar la correlación de los datos, los observatorios de la EVN envían diariamente información de la deriva de sus máseres de hidrógeno. La información se guarda en ficheros denominados `gps.xx`, donde `xx` es un indicativo de la estación. Estos ficheros contienen la media del offset del día anterior, el rms de los datos del día anterior y la media del día juliano modificado. Pueden contener más información siguiendo el formato descrito en el documento [3]. Los ficheros se almacenan en carpetas cuyo nombre indica el mes y año al que corresponden los datos.

El CAY envía diariamente un fichero llamado `gps.yb` al servidor FTP de la EVN `vlbeer.ira.cnr.it`. La generación del fichero la realiza un programa en Python denominado `gps_24.py`, que se muestra a continuación:

```
#!/usr/bin/env python

from smtplib import *
from math import *
from string import *
from ftplib import FTP
import sys, _mysql
#from time import *
from mx.DateTime import *
from commands import *

def main():
   ahora = now()
   hoy00 = ahora + RelativeDateTime(hour=0, minute=0, second=0)
   mjd_hoy = hoy00.mjd

   file_read = open('/home/maser/src/python/total.yb','r')
```

```
 data = file_read.readlines()
 nlin = len(data) -1
 last = data[nlin]
 last_mjd = atof(last[0:10])
 dif = int(ceil(mjd_hoy - last_mjd))
 file_read.close()

 for i in range(1,dif):
    dia00 = ahora + RelativeDateTime(days=-(dif-i), hour=0, minute=0, second=0)
    mjd_dia00 = dia00.mjd
    dia00str = dia00.strftime()
    mes = dia00str[4:7] + dia00str[22] + dia00str[23]
    dir = 'gps/' + lower(mes)
    dia23 = ahora + RelativeDateTime(days=-(dif-i), hour=23, minute=59, second=59)
    mjd_dia23 = dia23.mjd

    try:
       db = _mysql.connect(host = "localhost", user = "*******",
passwd = "*******", db = "******")
       except:
            print 'Error en la conexion a la base de datos'
            try:
               serv = SMTP()
               serv.connect('hercules.oan.es')
               subject = 'Error en la conexion a la base de datos'
               from_addr = 'maser@oan.es'
               to_addrs = ('a.garrigues@oan.es','p.devicente@oan.es')
               msg = "From: %s\r\nTo: %s\r\nSubject: %s\r\n" %(from_addr, to_addrs,
 subject) + 'No he podido conectarme a la base de datos'
               serv.sendmail(from_addr, to_addrs, msg)
               serv.quit()
               except:
                   pass
                   sys.exit(-1)
       query_str = "SELECT ts, mjd, maser_dps, rms FROM maserdata where mjd >= '"
+ str(mjd_dia00) + "' && mjd < '" + str(mjd_dia23) + "'"
       db.query(query_str)
       r = db.store_result()
       filas = r.fetch_row(0)
       filas2 = filas
       db.close()
       n = 0
       gps_aux = 0.0
       rms_aux = 0.0
       mjd_aux = 0.0
       d = 0.0
       file_total = open('/home/maser/src/python/total.yb','a')
       if dia00.day == 1:
            file_wr = open('/home/maser/src/python/gps.yb','w')
            file_wr.write("#   MJD     offset   rms   GPSname\n")
       else:
            file_wr = open('/home/maser/src/python/gps.yb','a')
       if len(filas) != 0:
```

```
            for f in filas:
               gps_db = atof(f[2])
               gps_aux += gps_db
               n +=1
            gps_med_24 = gps_aux/n
            for f2 in filas2:
               gps_db = atof(f2[2])
               mjd_db = atof(f2[1])
               d += (gps_db - gps_med_24)**2
               mjd_aux += mjd_db
            mjd_24 = mjd_aux/n
            rms_24 = sqrt(d/(n-1))
            if rms_24 == 0:
               rms_24 = 2.5e-10        #Error del contador 0.25 ns
            file_wr.write('%0.4f %0.4f %0.5f %s\n'  %(mjd_24, gps_med_24*1e6,
     rms_24*1e6, 'GPSYB1'))
            file_wr.close()
            file_total.write('%0.4f %0.4f %0.5f %s\n'  %(mjd_24, gps_med_24*1e6,
     rms_24*1e6, 'GPSYB1'))
            file_total.close()

            try:
               ftp = FTP('vlbeer.ira.cnr.it')
               ftp.login('***','***********')
            except:
               #print 'FTP: Error en la conexion'
               try:
                  serv = SMTP()
                  serv.connect('hercules.oan.es')
                  subject = 'FTP: Error en la conexion'
                  from_addr = 'maser@oan.es'
                  to_addrs = ('a.garrigues@oan.es', 'p.devicente@oan.es')
                  msg = "From: %s\r\nTo: %s\r\nSubject: %s\r\n" %(from_addr,
   to_addrs, subject) + 'No he podido conectarme al servidor FTP'
                  #serv.set_debuglevel(1)
                  serv.sendmail(from_addr, to_addrs, msg)
                  serv.quit()
               except:
                  pass
                  sys.exit(-1)
               try:
                  ftp.cwd(dir)
               except:
                  #print 'FTP: Error al cambiar de directorio'
                  ftp.close()
                  try:
                      serv = SMTP()
                      serv.connect('hercules.oan.es')
                      subject = 'FTP: Error al cambiar de directorio'
                      from_addr = 'maser@oan.es'
                      to_addrs = ('a.garrigues@oan.es', 'p.devicente@oan.es')
                      msg = "From: %s\r\nTo: %s\r\nSubject: %s\r\n" %(from_addr,
to_addrs, subject) + 'No he podido cambiar de directorio'
```

```
                        #serv.set_debuglevel(1)
                        serv.sendmail(from_addr, to_addrs, msg)
                        serv.quit()
                    except:
                        pass
                    sys.exit(-1)
                try:
                    ftp.storbinary('STOR gps.yb', open('/home/maser/src/python/gps.yb',
                except:
                    #print 'FTP: Error al guardar los datos'
                    ftp.close()
                    try:
                        serv = SMTP()
                        serv.connect('hercules.oan.es')
                        subject = 'FTP: Error al guardar los datos'
                        from_addr = 'maser@oan.es'
                        to_addrs = ('a.garrigues@oan.es', 'p.devicente@oan.es')
                        msg = "From: %s\r\nTo: %s\r\nSubject: %s\r\n" %(from_addr,
to_addrs, subject) + 'No he podido guardar los datos'
                        #serv.set_debuglevel(1)
                        serv.sendmail(from_addr, to_addrs, msg)
                        serv.quit()
                    except:
                        pass
                    sys.exit(-1)
                ftp.close()

if __name__ == "__main__":
        main()
```

El programa calcula el día juliano modificado correspondiente al momento actual, lee el último día juliano modificado almacenado en el fichero `total.yb`, y rellena los ficheros `gps.yb` y `total.yb` con los datos correspondientes a los días transcurridos entre el último dato almacenado en `total.yb` y el día actual empleando la información guardada en la base de datos. Para cada día, lee el día juliano modificado (mjd) y offset almacenados en la tabla `maserdata` de la base de datos, calcula la media de los dias julianos modificados, la media y el rms del offset y añade una línea con esta información en los archivos `gps.yb` y `total.yb`. Si es día primero de mes en lugar de añadir una línea al archivo `gps.yb` se crea un archivo nuevo con ese mismo nombre. Esta aplicación rellena más de un archivo `gps.yb` de cada mes si fuese necesario. Es decir si desde el momento que se creó o rellenó por última vez el archivo `gps.yb` ha transcurrido más de un mes, el programa generará o/y rellenará los archivos `gps.yb` correspondientes a los días y meses que faltan. Para poder realizar esta operación en el archivo `total.yb` se mantiene la media diaria desde que se comenzó a monitorizar el máser.

Para poder almacenar el fichero `gps.yb` en su carpeta correspondiente también se calcula el mes y año de ese día. El programa hace una conexion FTP, entra en el directorio correspondiente al mes y al año y transfiere allí el fichero `gps.yb`. Si se produce algún problema en la conexión FTP o en la lectura de la base de datos se genera un mensaje de correo electrónico de aviso a *p.devicente@oan.es* y a *a.garrigues@oan.es*. Este programa se repite ciclicamente todos los

días a las 02:05 horas empleando un “Crontab”.

Como se acaba de explicar si falla la transferencia o el registro de los datos, el programa es capaz de transferir al servidor todos los datos que faltan.

6. Evolución del máser KVARZ CH1-75 instalado en Yebes entre los años 1998 a 2004

6.1. Variación del flujo de hidrógeno molecular y detector de fase con el tiempo

Como ya se ha mencionado en una sección anterior, cada mes se guarda la información del estado del máser. Se ha observado que algunos de sus parámetros muestran una deriva sistemática con el tiempo, como es el caso del flujo de hidrógeno molecular y de la fase detectada. En las figuras 4 y 5 se muestan las gráficas de la dependencia temporal del detector de fase y del depósito de hidrógeno molecular junto con la de la temperatura de la sala. El depósito de H_2 es un compuesto químico sólido que genera H_2 de modo contínuo.

Para determinar si el flujo de H_2 guarda cierta dependencia con la temperatura de la sala, se ha pasado una línea de base a la gráfica del hidrógeno molecular de la figura 4 empleando una línea recta a partir de 2 puntos seleccionados en los que la temperatura es la misma. La figura 5 muestra la variación de H_2 una vez sustraida un línea de base, junto con la temperatura. Aparentemente se aprecia una pequeña correlación que indica que el flujo de H_2 aumenta ligeramente con la temperatura del entorno del máser.

6.2. Variación de la frecuencia del máser con el tiempo

El contador GPIB mide el tiempo transcurrido entre un pulso generado por el sistema GPS y un pulso generado por el máser. El pulso se genera en cada equipo a partir de una señal periódica de 5 MHz y tras contar un número determinado de ciclos. Si la frecuencia de ambos sistemas difiere, el tiempo transcurrido entre ambos pulsos se incrementa de modo proporcional a la diferencia de frecuencia. Sean V_{GPS} y V_{MASER} dos señales periódicas generadas por el GPS y el máser respectivamente.

$$\begin{aligned} V_{GPS}(t) &= \cos(2\pi\nu t) \\ V_{MASER}(t) &= \cos(2\pi\nu t + \phi_2(t)) \end{aligned}$$

Como suponemos que ambos equipos generan una señal de 5 MHz la fase residual de la señal del máser se puede considerar fruto de una diferencia de frecuencia entre ambos más un término constante

$$V_{MASER}(t) = \cos(2\pi\nu t + 2\pi\delta\nu t + \phi_0)$$

Sea δt el tiempo transcurrido desde que la fase de la señal 1 toma un valor hasta que la fase de la señal 2 toma el mismo valor

$$2\pi\nu t = 2\pi\nu(t + \delta t) + 2\pi\delta\nu(t + \delta t) + \phi_0$$

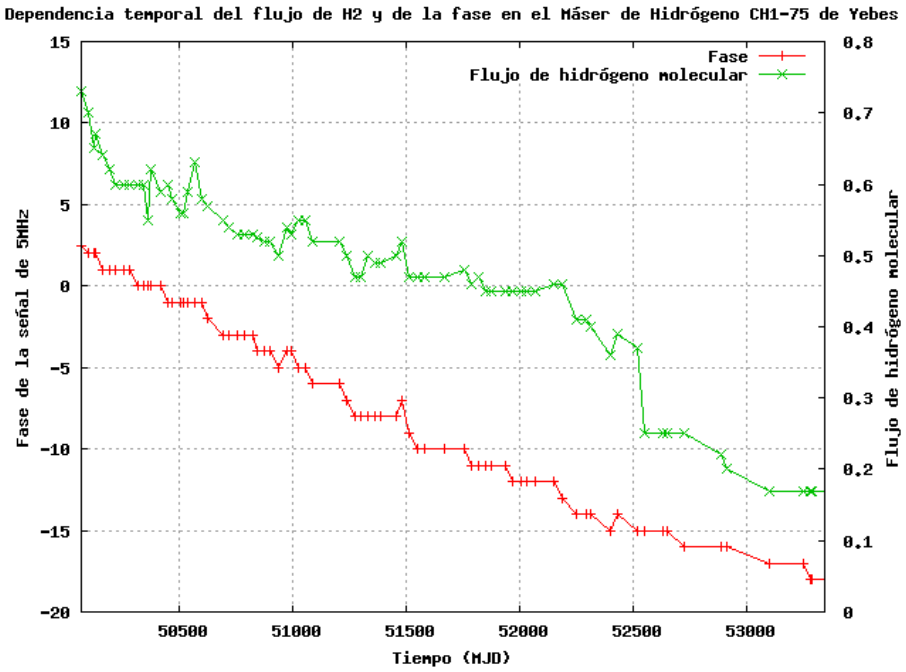

Figura 4: *Estado del depósito de H molecular (línea en verde) y del detector de fase (línea en rojo) desde diciembre de 1995 hasta diciembre de 2004*

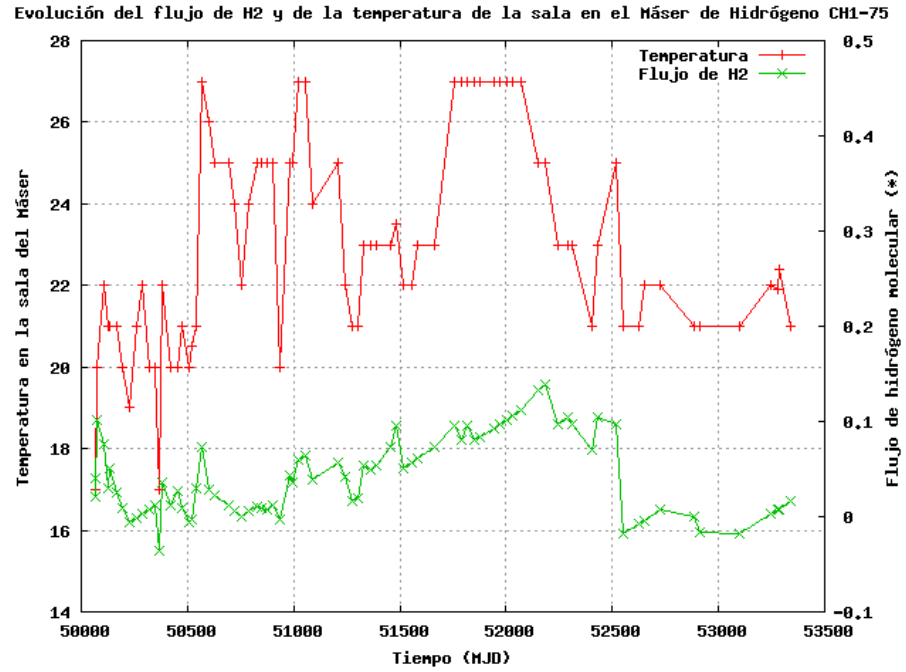

Figura 5: *Temperatura en la sala del máser y flujo* de H_2 desde de diciembre de 1995 hasta diciembre de 2004. (*) H_2 una vez pasada la línea linea de base: f(t) =* $-1{,}4522\,10^{-4}\,t + 7{,}9$

entonces,

$$-2\pi(\nu+\delta\nu)\delta t = 2\pi\delta\nu t + \phi_0$$

y

$$\delta t = -\left(\frac{\delta\nu}{\nu+\delta\nu}t + \frac{\phi_0}{2\pi(\nu+\delta\nu)}\right) \tag{2}$$

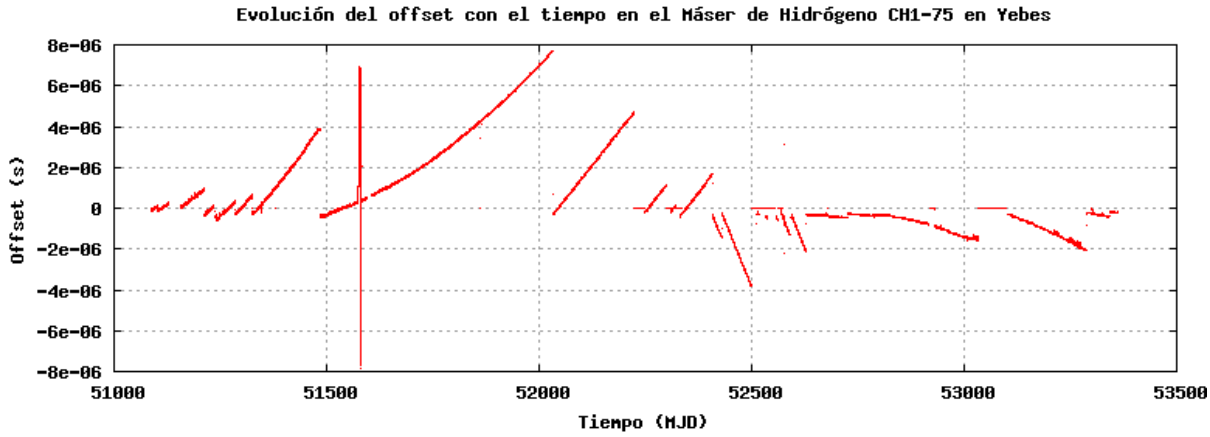

Figura 6: *Error relativo en frecuencia del máser desde octubre de 1998 hasta diciembre de 2004*

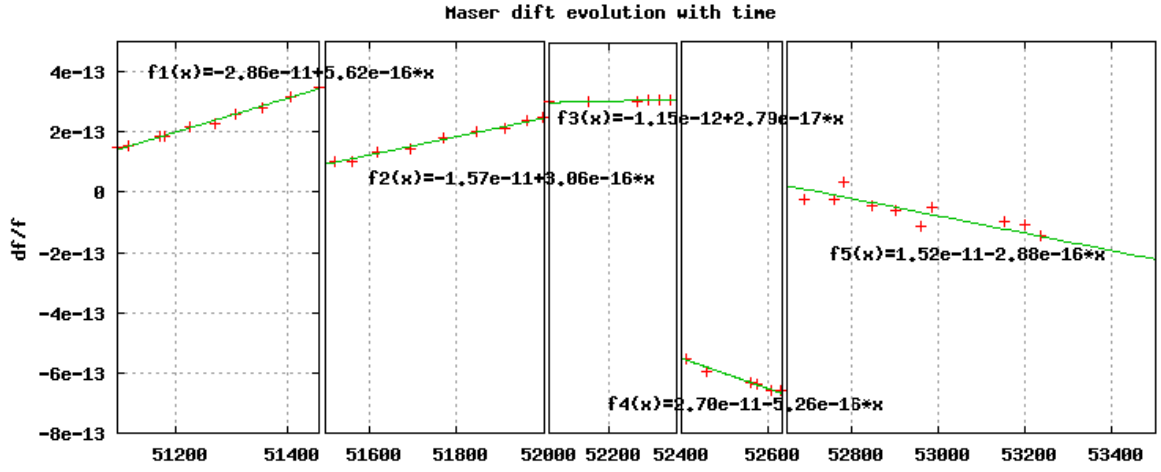

Figura 7: *Deriva del error relativo en frecuencia del máser desde octubre de 1998 hasta diciembre de 2004. En el eje de abscisas día julano modificado*

Por tanto el tiempo transcurrido (o diferencia de tiempo entre pulsos) es proporcional al error relativo en frecuencia y se comporta como una línea recta donde la pendiente es el error

relativo en frecuencia y la ordenada depende de la frecuencia y fase inicial de una de las señales. Cuando

$$\nu_{MASER} > \nu_{GPS} \quad \Rightarrow \delta\nu > 0 \tag{3}$$

δt tendrá una pendiente negativa y cuando

$$\nu_{MASER} < \nu_{GPS} \quad \Rightarrow \delta\nu < 0 \tag{4}$$

la pendiente de δt será positiva. Es decir cuando el máser oscila más lento que el GPS los dos pulsos se separan y la diferencia de tiempos entre el pulso del GPS y del máser es creciente. En este momento la pendiente que presenta el máser del CAY es positiva (Fig. 9) pero los valores son negativos, es decir la señal de 5MHz del máser va retrasada con respecto a la del GPS y su frecuencia es ligeramente menor que la del GPS.

La figura 6 muestra la evolución de la diferencia de tiempos desde octubre de 1998 hasta finales de 2004. La figura 7 muestra, para este mismo periodo, la evolución de su deriva, es decir, la variación de la diferencia de tiempos con el tiempo, o la derivada de la frecuencia relativa. Como se puede apreciar en la figura 6 la diferencia de frecuencias entre el máser y el sistema GPS varía con el tiempo, probablemente por variaciones en la cavidad del máser. Las discontinuidades en las gráficas se originan cuando se resincronizan ambas señales tras ajustar la cavidad del máser empleando un varactor. Cuando el varactor haya empleado todo su recorrido será necesario ajustar mecánicamente la cavidad y redifinir el cero del varactor. Los errores relativos en frecuencia típicos son iguales o inferiores 10^{-13} Herzios por Herzio, o si empleamos una escala temporal, 10^{-13} segundos por segundo. El cuadro 3 muestra los errores relativos en frecuencia para las fechas en las que se sintonizó la cavidad. La sintonía de la cavidad se hizo en esos casos para compensar el error relativo en frecuencia determinado a partir de los días previos.

Fecha	$\Delta f/f$	Varactor antes del cambio	Varactor después del cambio
12-03-1997	$2,10\,10^{-13}$	3272	3126
01-04-1998	$2,95\,10^{-13}$	3126	2737
29-10-1999	$3,50\,10^{-13}$	2737	2533
07-05-2002	$0,91\,10^{-12}$	2533	1707
19-12-2002	$-6,36\,10^{-13}$	1707	2324
01-10-2004	$-1,39\,10^{-13}$	2324	2414
11-03-2005	$-2,25\,10^{-14}$	2414	——

Cuadro 3: *Errores relativos en frecuencia determinados antes de realizar un ajuste de la cavidad empleando el varactor. También se incluye la tensión del varactor antes y después de los cambios.*

La figura 7 muestra que el error relativo en frecuencia varía en función del tiempo en una cantidad de 10^{-16} Herzios por Herzio y por segundo.

7. Estudio del mejor tiempo de integración para obtener el error relativo en frecuencia

El contador mide cada segundo el tiempo transcurrido entre los dos pulsos. Considerando la señal del GPS como referencia, el intervalo de tiempo medido (offset a partir de ahora), mide la fase de la señal del máser. Obviamente la señal del GPS no se puede considerar una referencia ideal ya que tiene mucho mayor ruido de fase que la señal generada por el máser pero en la práctica la consideraremos una señal ideal. Si suponemos por tanto que el pulso del GPS marca el momento en el que su fase toma un valor (por ejemplo podemos considerar que sea 0), el segundo pulso indica el momento en el que la señal de 5 MHz del máser toma el mismo valor. Por tanto en el intervalo de tiempo δt_k (offset) la fase de la señal habrá cambiado,

$$\phi_k = 2\pi\nu_0\delta t_k \tag{5}$$

donde $\nu_0 = 5$ MHz y δt_k es el offset medido con el contador en el momento k.

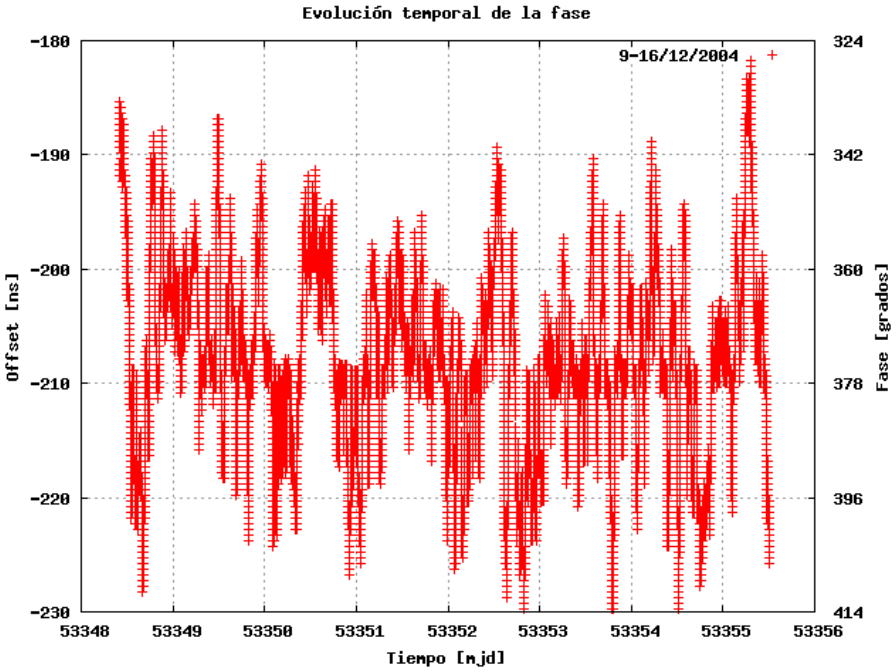

Figura 8: *Diferencia temporal entre los pulsos del GPS y del máser para un intervalo de 8 días 9-16/12/2004. En el lado izquierdo de la gráfica se muestra la fase de la señal para facilitar la conversión al lector. La fase se ha calculado para una señal de 5 MHz*

La figura 8 muestra el offset medido cada segundo durante ocho días contíguos. La escala del eje de ordenadas izquierdo muestra el offset en nanosegundos y la escala del eje de ordenadas derecho la fase en grados. Como se puede observar la fase varía de modo errático entre 180 y 230 nanosegundos y en los ocho días no se puede apreciar fácilmente una tendencia que indique una frecuencia residual. Las variaciones de fase con picos no son características de ruido blanco

y con toda seguridad son debidas al sistema de recepción de la señal del GPS. Si se tratase de ruido blanco deberíamos esperar una nube de puntos en torno a una variación sistemática de fase causada por una frecuencia residual.

De acuerdo con los datos obtenidos cada 10 minutos durante los 3 meses siguientes a la días de la gráfica de la figura 8 el error relativo en frecuencia del máser era en ese momento inferior a 10^{-14} (ver la figura 9). Esto quiere decir que en ocho días el offset debería cambiar $10^{-14} \cdot 8 \cdot 86400 \cdot 10^9 \simeq 7$ ns, cantidad inapreciable en la gráfica de la figura 8.

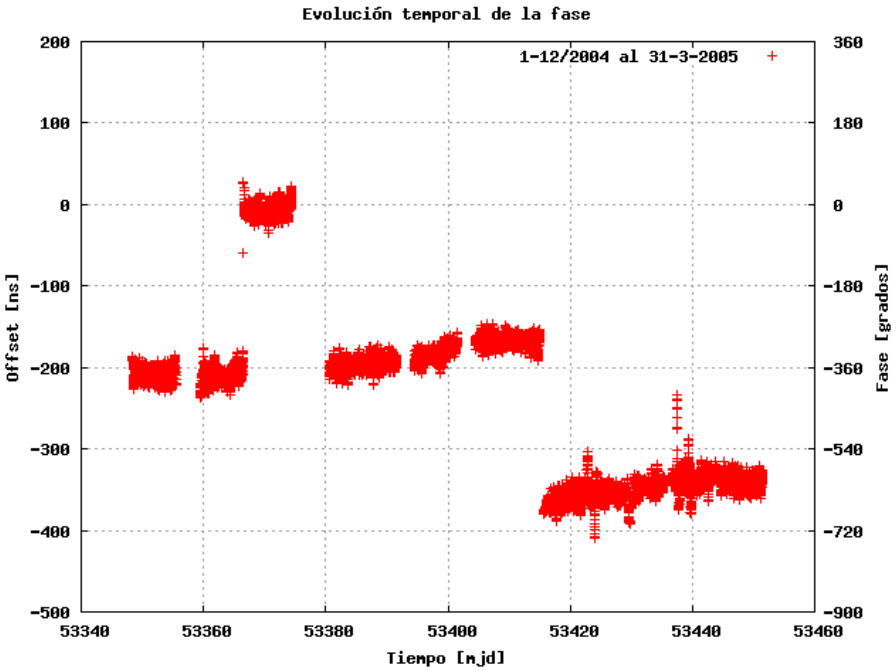

Figura 9: *Diferencia temporal entre los pulsos del GPS y del máser desde el día 9-12-2004 hasta el 22-3-2005. En el lado izquierdo del eje de ordenadas se muestra la fase de la señal para facilitar la conversión al lector. La fase se ha calculado para una señal de 5 MHz*

En la figura 10 se ha representado la diferencia temporal durante los primeros 10 minutos de la figura 8. Como se puede observar la dispersión casi instantánea es de 2 nanosegundos o 4 grados pico a pico aproximadamente y los datos muestran saltos debido a al precisión del contador (0,25 ns).

Como ya se ha mencionado en una sección anterior los datos se almacenan en una base de datos cada 10 minutos. Para estimar el tiempo de integración en el cual el ruido de la fase se hace mínimo hemos calculado el error cuadrático medio para diferentes intervalos de integración. El modo de cálculo es el siguiente:

$$\sigma(\tau) = \sqrt{\frac{\sum_k (\delta t_k(\tau) - \overline{\delta t})^2}{n}}$$

donde $\overline{\delta t}$ es el valor medio del offset entre los pulsos y se obtiene promediando todos ellos,

$$\overline{\delta t} = \frac{\sum_k \delta t_k}{n}$$

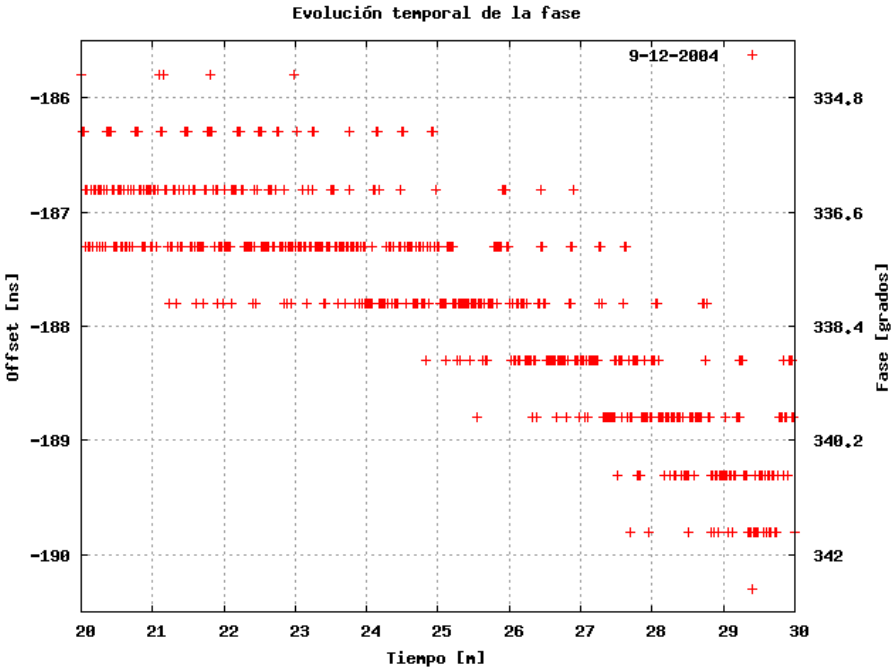

Figura 10: *Diferencia temporal entre los pulsos del GPS y del máser para los primeros 10 minutos de la gráfica 8. En el eje de ordenadas se muestra la fase de la señal para facilitar la conversión al lector. La fase se ha calculado para una señal de 5 MHz*

y donde $\delta t_k(\tau)$ se calcula del siguiente modo:

$$\delta t_k(\tau) = \frac{\sum_{i=1}^{N} \delta t_i}{N} \tag{6}$$

siendo $N = \tau/T$ y T el periodo con el que el contador toma datos, es decir, 1 segundo, y δt_n es el resultado obtenido por el contador para el momento n. Por tanto los datos se agrupan de 1 en 1, de 2, en 2, de 3 en 3, y así sucesivamente y se calcula el error cuadrático medio obtenido para cada agrupamiento o tiempo de integración. En la figura 11 se ha representado σ_τ para diferentes tiempos de integración (τ) y el mismo intervalo de datos de los 8 días de la figura 8.

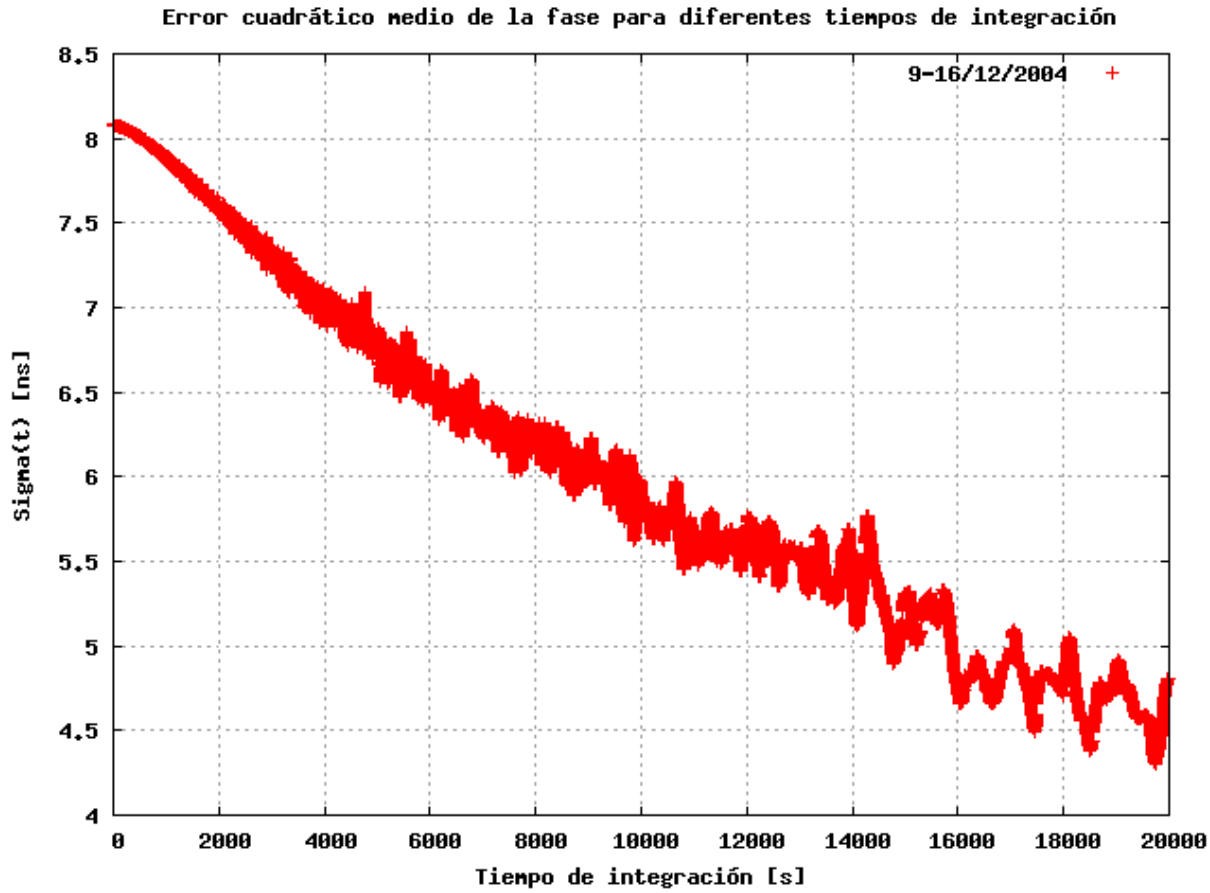

Figura 11: *Diferencia temporal promediada entre los pulsos del GPS y del máser para diferentes tiempos de integración realizados entre los días 9-12-2004 y 16-12-2004. La fase se ha calculado para una señal de 5 MHz*

El error cuadrático medio de la fase disminuye con el tiempo de integración y muestra oscilaciones cuando el tiempo de integración crece. Probablemente estas oscilaciones se corresponden con las oscilaciones de fase que se observan en la figura 8. Es de esperar que si la deriva en frecuencia del máser fuese mayor, el error cuadrático medio aumentara a partir de ciertos valores del tiempo de integración. Si este cálculo se hubiese hecho para un intervalo de

medio día el error cuadrático medio de la fase habría mostrado un comportamiento completamente diferente causado por las fuertes oscilaciones de la fase que se observan.

De este estudio se concluye que no hay un mejor tiempo de integración y que este depende fuertemente de la deriva en frecuencia del máser para ese periodo. Por tanto consideramos razonable obtener el promedio del offset temporal cada 10 minutos ya que esta operación no afecta de modo significativo al cálculo de la deriva en frecuencia del máser de hidrógeno. El valor de 10 minutos se eligió para un contador con una resolución mucho peor considerando que el número de muestras por día (144) no es excesivo y permite cierta resolución a lo largo del día.

8. Estudio de la estabilidad de fase del sistema GPS - máser

La estabilidad de fase de los máseres se estudia empleando dos equipos similares, e introduciendo las señales de ambos en un comparador de fase, que habitualmente multiplica por un cierto factor la amplitud de la diferencia de fase. El comparador produce una señal proporcional a la diferencia de fase de ambos osciladores. La constante de proporcionalidad se determina introduciendo una diferencia de frecuencia conocida y midiendo la amplitud de la sinusoide resultante. Se puede ver con detalle un ejemplo de este montaje en el cuaderno de registro del máser y en el informe IT-OAN-2005/6. Sin embargo en esta seccion intentaremos extraer información sobre el ruido de fase del sistema GPS - máser a partir de las medidas del tiempo transcurrido entre los pulsos del GPS y del máser.

La estabilidad en frecuencia de un oscilador se puede estimar midiendo la varianza del error relativo en frecuencia $y(t)$. Es decir es una medida de la dispersión de la fase normalizada por el valor de la frecuencia, de modo que dicha dispersión no dependa de la frecuencia de la señal.

La fase de un oscilador se puede expresar del siguiente modo:

$$\phi = 2\pi\nu_0 t + \phi(t) \tag{7}$$

La frecuencia instantánea de esta señal se define como:

$$\begin{aligned} \nu(t) &= \frac{1}{2\pi}\frac{d(2\pi\nu_0 t + \phi(t))}{dt} && (8)\\ &= \nu_0 + \frac{d\phi(t)}{2\pi dt} && (9)\\ &= \nu_0 + \delta\nu(t) && (10) \end{aligned}$$

y el error relativo en frecuencia se define como:

$$y(t) = \frac{\delta\nu(t)}{\nu_0} = \frac{d\phi(t)}{2\pi\nu_0 dt} \tag{11}$$

El error relativo en frecuencia se puede medir examinando como varía la fase entre dos instantes diferentes separados temporalmente una cantidad τ. Se puede demostrar (ver [5]) que la cantidad normalizada $\overline{y_k}$,

$$\overline{y_k} = \frac{1}{\tau}\int_{t_k}^{t_k+\tau} y(\theta)d\theta$$

en el momento k vale:

$$\overline{y_k(\tau)} = \frac{(\phi(t_k+\tau)-\phi(t_k))}{2\pi\nu_0\tau} \tag{12}$$

donde los instantes de medida $k = 1, 2, 3, ...$ están separados por un tiempo $T \geq \tau$,

$$t_{k+1} = t_k + T \tag{13}$$

La estabilidad en frecuencia es entonces la varianza de $\overline{y_k}$

$$\langle\sigma_y^2(N,T)\rangle = \frac{1}{N-1}\langle\sum_{n=1}^{N}(\overline{y}_n - \frac{1}{N}\sum_{k=1}^{N}\overline{y_k})^2\rangle \tag{14}$$

La varianza de Allan es un caso particular del anterior en el que $N = 2$ y $T = \tau$,

$$\sigma_y^2(T) = \frac{\langle(\overline{y_{k+1}} - \overline{y_k})^2\rangle}{2} \tag{15}$$

Como el contador de tiempos mide una diferencia de tiempos,

$$\delta t_k = \frac{\phi(t_k)}{2\pi\nu_0} \tag{16}$$

cada 1 s y el error relativo en frecuencia vale,

$$\overline{y_k} = \delta t_{k+1} - \delta t_k \tag{17}$$

y la varianza de Allan es,

$$\sigma_y^2 = \frac{1}{2N}\sum_{k=1}^{N}(\delta t_{k+1} - 2\delta t_k + \delta t_{k-1})^2 \tag{18}$$

La figura 12 muestra la desviación típica de Allan para diferentes tiempos de integración múltiplos del periodo entre muestras (T, 2T, 3T ...) y para los datos representados en la figura 8.

Como se puede apreciar, a medida que aumenta el tiempo de integración disminuye la varianza. Además la dispersion es mayor con mayores tiempo de integración porque disponemos de menos datos con los que realizar el cómputo. En esta gráfica además se han representado la varianza para el máser de hidrógeno KVARZ CH1-75, un oscilador de Cesio de alta estabilidad y el receptor GPS Truetime XL-DC instalado en el Centro Astronómico de Yebes.

En la figura 12 se puede apreciar que la varianza de Allan calculada es ligeramente mejor que la que se supone teóricamente para el terminal GPS XL-DC de TrueTime, por lo que consideramos que la varianza observada procede fundamentalmente del sistema de recepción GPS. Probablemente las especificaciones del receptor GPS de TrueTime se obtuvieron con un sistema de medida similar al nuestro pero empleando un oscilador de Cesio. Si esta hipótesis es correcta nuestras medidas serían mejores porque se ha usado como oscilador local un máser de hidrógeno lo que reduce en un orden de magnitud los valores especificados.

No es posible por tanto medir la varianza de Allan del máser empleando este sistema, aunque si es posible medir la deriva en frecuencia examinando el comportamiento a muy largo plazo. Cuanto menor es la deriva, como en este caso, mayor tiempo de integración es necesario para conocer esta.

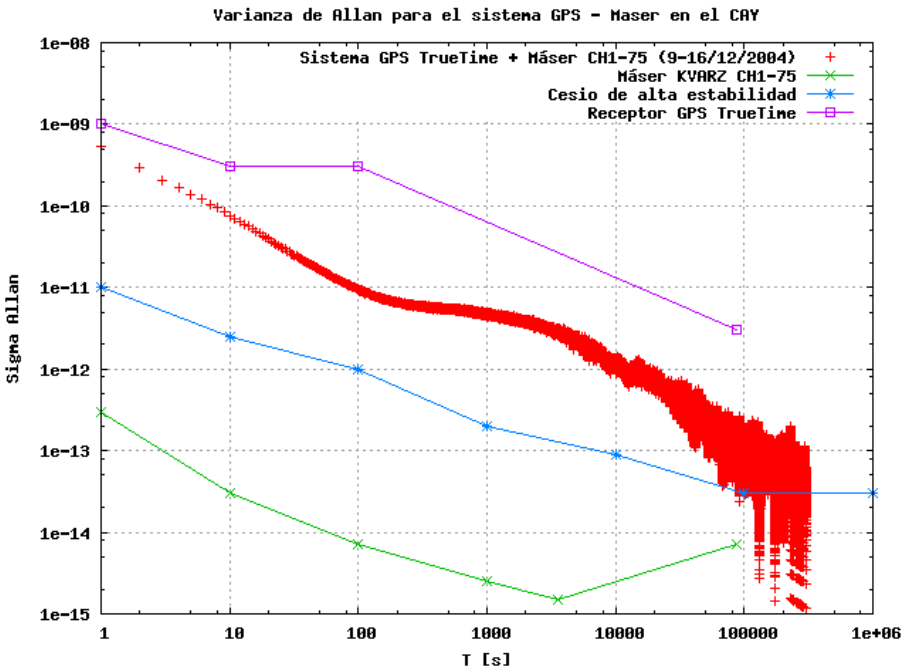

Figura 12: *Desviacion típica de Allan del sistema GPS - máser en función del tiempo para los datos de la figura 8. También se han representado las varianzas para el máser CH1-75, un Cesio de alta estabilidad y para el sistema GPS TrueTime del que diponemos en Yebes*

Referencias

[1] Beazley D.M., *Python Essential Reference*, New Riders.

[2] DuBois P., *MySQL*, Developer's Library.

[3] van Langevelde H. J., *GPS clocks in th EVN; Toward blind correlation*, EVN, Informe Técnico 65, 1996.

[4] Prata S., *C++ Primer Plus*, The Waite Group's.

[5] Rutman J., Proceedings of the IEEE, 66, 9, 1978.

[6] Stevens W.R., *UNIX Network Programming*, Pretice Hall Software Series.

[7] Thompson A.RMoran J. M., Swenson,JR G.W., *Interferometry and Synthesis in Radio Astronomy*, Krieger Publishing Company.

[8] de Vicente P., López I., *Marcas temporales en el sistema VLBI del CAY*, Informe Técnico IT-OAN 2001-8.

[9] de Vicente P., *Programas de control y monitorización de la estación meteorológica y del terminal GPS del CAY*, Informe Técnico IT-OAN 2002-2.