

IRAM-COMP-069

Revision: 1 2013-06-13

Contact Author

# Institut de RadioAstronomie Millimétrique

# HEMT software



Owner Sebastien BLANCHET

Keywords: hemt, software

# Change Record

REVISION	DATE	AUTHOR	SECTION/PAGE AFFECTED	REMARKS
0	2010-08-18	Blanchet		Initial release
1	2013-06-13	Blanchet	Many	Update for debian 7.0

# Content

1 Introduction	<u>4</u>
2 Main features	4
3 Installation	4
3.1 Requirements	4
3.2 Installation instructions for requirements	<u>5</u>
3.3 Build HEMT software	<u>5</u>
3.4 Configure environment	6
3.5 Initial test	6
3.6 Start application automatically	7
4 Internal programs	<u>7</u>
4.1 CanManager	7
4.2 HEMT Server	7
4.2.1 Syntax	8
4.2.2 API Description	8
4.3 XML RPC client examples	10
4.3.1 Minimal python example	10
4.3.2 Python examples	10
4.3.3 C++ client example	10
4.4 hemt-init-ampli	10
4.4.1 Syntax	10
4.5 hemt-check-ampli	10
4.5.1 Syntax	11
4.6 Simulator	11
4.6.1 Main window	11
4.6.2 LO Simulation Window	11
4.6.3 Ampli Simulation Window	<u>13</u>
4.6.4 Cryo Simulation Window	14
5 User Programs	<u>15</u>
5.1 hemt-i2c-reset	15
5.1.1 Syntax	15
5.1.2 Example	15
5.2 UtilCan	16
5.2.1 Syntax	16
5.2.2 Screenshot	16
5.3 CanLogger	16
5.3.1 Syntax	16
5.3.2 Example	17
5.4 Lo	17
5.4.1 Syntax	17
5.4.2 Example	17
5.5 Gui	18

5.5.1 Main window	<u>8</u>
5.5.2 Lo window1	9
5.5.3 Amplifiers window2	1
5.5.4 Cryo window	2
5.6 Rop	2
5.7 hemt-GetStatus	3
5.7.1 Syntax	3
5.7.2 Example	3
5.8 Check software	4
5.8.1 Syntax	4
5.8.2 Example	4
5.9 hemt-SetProtection.pv	4
5.9.1 Svntax	5
5.9.2 Example	5
5.9.3 Alternative methods	5
<u>6 Daily Operation2</u>	<u>5</u>
6.1 Modify database configuration 2	5
6 2 L O data files	6
	2
<u>7 Tips</u> 2	6
7.1 How to change the fonts size ?	6
8 Troubleshooting	7
8.1 IPv6	7
8.2 Procedure to reset the I2C bus	7
	_

# 1 Introduction

In 2010, IRAM has installed a new prototype receiver called HEMT at the IRAM 30M telescope. This document is the documentation reference for the HEMT software: installation, technical documentation, daily usage and troubleshooting.

# 2 Main features

- The software is written in C++/Qt.
- Two specific graphical interfaces for laboratory and observing.
- Automatic lo tuning.
- Built-in Simulator.
- XML RPC server for easy integration with observing software.
- Embedded SQL database to store configuration.

# 3 Installation

# 3.1 Requirements

# Hardware

The minimal requirements for hardware are:

- CPU: x86 CPU at 500 MHz
- RAM: 512 MB
- CAN controller: TPMC816 PMC card from Tews Technologies GmbH

For software development, the CAN controller is optional, because the software provides a CAN simulator.

# **Operating system**

The software targets Linux Debian 5.0, 6.0 or 7.0 amd64 as main platformw, but it should run on any computer with a recent Linux version.

# Mandatory libraries

The following libraries are required to build the software

Name	Description	Version	Download
g++	GNU C++ compiler	>= 4.1	http://gcc.gnu.org
subversion	Subversion is an open source version	>= 1.5	http://subversion.tigris.org
	control system.		
Qt	C++ Cross-platform application	>= 4.4.3	http://www.qtsoftware.com
	framework		
Sqlite	Embedded SQL database	>= 3.3.6	http://sqlite.org
Xmlrpc-c	XML-RPC for C and C++.	>= 1.06	http://xmlrpc-c.sourceforge.net

All these libraries are very common, and you should find easily ready-to-install packages for your favorite Linux distribution.

# **Recommended software**

The following programs are strongly recommended to modify easily the code.

Name	Description	Version	Download

Eclipse/CDT	С	and	C++	Integrated	Development	>= 3.5	http://eclipse.org/cdt
	Env	vironme	ent (IDE	) for the Eclip	ose platform.		
doxygen	Automatic documentation system			m	>= 1.5.6	http://doxygen.org	

All these software are very common, and you should find easily ready-to-install packages for your favorite Linux distribution.

# 3.2 Installation instructions for requirements

This procedure explains how to install the HEMT receiver software requirements on a fresh Debian 5.0 (Lenny) installation.

Install Linux Debian 7.0 amd64 on a computer.

Install development tools and libraries

```
# aptitude install rsync gcc g++ doxygen make gnuplot
# aptitude install manpages-dev graphviz sqlite3
# aptitude install libqt4-dev qt4-doc-html libqt4-sql-sqlite \
libxmlrpc-c3-dev qt4-qtconfig libcurl4-openssl-dev subversion
```

Note: the Linux headers are required to build the CAN driver: # aptitude install linux-headers-`uname -r`

Install useful packages:

```
# aptitude install openssh-server nmap xxdiff sqlite3-doc
```

Create the oper account

\$ su -c "adduser oper"

# **3.3 Build HEMT software**

#### Extract code from the repository

```
$ mkdir ~/develSVN
$ cd ~/develSVN
$ svn co svn://svn.iram.fr/30M/hemt/trunk hemt
```

Then use the Makefile to extract automatically the dependencies

- \$ cd hemt \$ qmake
- \$ make get\_deps

Build the driver for the TPMC816 CAN board (not required if you use only the simulation)

```
$ cd ~/develSVN/tdrv011
$ su
# make install
```

Create a group which is allowed to access /dev/tdrv011\_\* devices

```
# groupadd can
# echo 'KERNEL=="tdrv011_[0-9]*", MODE="0664", GROUP="can"' > \
/etc/udev/rules.d/99-CAN.rules
```

On debian 7.0, the driver is loaded automatically, so there is nothing special to do.

#### Build the hemt code

Note: If you do not use Debian, you have to modify firstly the file Utils/conf/conf.pri to specify the libraries locations

\$ cd ~/develSVN/hemt
\$ ./build.sh

Optionally, you can build the API documentation. The documentation will be created in the doxydoc subdirectory.

\$ make doc

Install the CAN tools

```
$ su
# cd ./CanIp/
# ./install.sh programs
# exit
```

Then, install the software **and the default data** in /home/introot/hemt Note: the shared libraries are installed in /home/introot/lib

```
# cd /hemt
#./install.sh all
```

#### Warning:

To update only the software without reinstalling the default settings, use # make -f Makefile.install programs

Nevertheless it is safer to backup /home/introot/hemt first, so you can restore the original installation in case of errors.

# tar cvfz ~/hemt-backup-`date --iso`.tar.gz /home/introot/{hemt,lib}

#### 3.4 Configure environment

You should add /home/introot/hemt/bin to your path

To decode CAN identifier into symbolic names in CanLogger, you should also define CAN\_DB\_FILE as follow

```
export DEVICE_NAME=hemt
export CAN_DB_FILE=${INTROOT}/data/${DEVICE_NAME}.db
```

#### 3.5 Initial test

For this initial test, we run 1.The CanManagers to enable the Can/IP protocol 2.The simulator 3.The autotuning lo program

First we remove the /dev/tdrv011\_\* devices, so the CanManager will run in simulation mode

```
$ su -c "rm -f /dev/tdrv011_*"
$ hemt-init-can.sh
$ xterm -e hemt-simul &
```

(click on LO button to activate the LO simulation)

Now run the LO autotuning program: **\$ hemt-lo -f=70.0** 

```
Creation date: April 2010
HEMT Software
```

# 3.6 Start application automatically

Add /home/introot/hemt/bin/hemt-init-receiver.sh to /etc/rc.local to initialize the software at the startup.

# 4 Internal programs

This section describes internal programs that drive the receiver. These programs are executed automatically, therefore normal users are not expected run them.

# 4.1 CanManager

For a complete description see the document can-ip.pdf

```
$ CanManager -h
CanManager is a bridge between the CAN bus and the CAN/IP protocol
Usage: CanManager [options]
Options:
  -d=/dev/devname
                      CAN controller device name to use. If missing,
                        the application runs in simulation mode
  -p=udpPort
                      Listen to UDP port udpPort
  -t1=delay1_us
                      Delay in microseconds between two CAN 1.0 messages.
                          Default=0
  -t2=delav2 us
                      Delay in microseconds between two CAN 2.0 messages.
                          Default=0
                 Display version information
     -?
                 Display help
  -h,
Example:
        CanManager -d=/dev/tdrv011_0 -p=2500 -l=20
```

For the HEMT software, CanManager must listen on port udp/2501
CanManager -d=/dev/tdrv011\_1 -p=2501 -t1=100000

# 4.2 HEMT Server

HEMT server is an XML-RPC server to remotely control the receiver.

# What is XML RPC ?

XML-RPC is a remote procedure call protocol that uses XML to encode its calls and HTTP as a transport mechanism. For a detailed introduction to XML RPC see <a href="http://en.wikipedia.org/wiki/XML-RPC">http://en.wikipedia.org/wiki/XML-RPC</a>

This protocol is very simple to use, and can be used from any programming language (many opensource libraries are available)

HEMT-server listens for XML-RPC calls on <a href="http://localhost:1080/RPC2">http://localhost:1080/RPC2</a>

Note: The path /*RPC2* is the default path for a XML-RPC server. Therefore, it can be sometimes omitted (it depends on the implementation library)

This server supports: •introspection <u>http://xmlrpc-c.sourceforge.net/introspection.html</u> •multicalls

#### 4.2.1 Syntax

```
$ hemt-server -h
HEMT Server - XML-RPC Server
Listen on port 1080
Usage: hemt-server [options]
Options:
  -v Display version information
  -h, -? Display help
```

# 4.2.2 API Description

Since the XML-RPC server support introspection, we can retrieve the API with a simple program

```
$ ./ListMethods.py
ampli.setProtection
receiver.getCalibration
receiver.getPositionList
receiver.getStatus
receiver.setAttenuator
receiver.setCalibration
receiver.setCalibration
receiver.setLoSwitch
receiver.tuneLo
system.listMethods
system.methodHelp
system.methodSignature
system.multicall
system.shutdown
```

```
$ ./Introspection.py
# Connect to http://localhost:1080
-----
                              Name : ampli.setProtection( string, int )
Return Type: int
Description: Set amplifier protection.
Return code is always zero.
Syntax: ampli.setProtection( string ampliName, int protection)
 - valid ampli names are: all, ampli_H1, ampli_H2, ampli_V1, ampli_V2
 - protection: 0 => Off, 1 => On
-----
Name : receiver.getCalibration( )
Return Type: string
Description: Get current calibration position
Return string: moving, off, sky, hot, cold
Syntax: GetCalibration( )
: receiver.getPositionList( )
Name
Return Type: array
Description: Get position list for receiver.SetCalibration()
Return Array of string.
Syntax: GetPositionList( )
Name : receiver.getStatus( )
Return Type: struct
Description: Returns the receiver status
```

```
: receiver.setAttenuator( string, int )
Name
Return Type: int
Description: Set attenuator. Return code is always zero.
Syntax: SetAttenuator( string attenuatorName, int attenuationDecibel)
- 'attenuatorName' in { 'H', 'V' }
- 'attenuationDecibel' must be in range [ 0, 31.5 ]
-----
     : receiver.setCalibration( string )
Name
Return Type: int
Description: Set Calibration.
Return code: - 0 : OK
   - 1 : Error
Syntax: SetCalibration( string positionName )
Valid position names are:
off, sky, hot, cold
-----
Name : receiver.setLoSwitch( string, int )
Return Type: int
Description: Set a LO switch.
Return code is always zero.
Syntax: setLoSwitch( string switchName, int switchValue)
- switchName must be in: { gunn, deltaf, loop, sweep }
- switchValue must be 0 or 1
-----
Name
     : receiver.tuneLo( double, string )
Return Type: int
Description: Tune LO.
Return code: - 0 : OK
   - 1 : Error
Syntax: tuneLo( double loFreq, string deltaF )
   - loFreq: frequency in GHz
   - deltaF: '-' or '+'
                          .....
Name : system.listMethods( )
Return Type: array
Description: Return an array of all available XML-RPC methods on this server.
-----
       : system.methodHelp( string )
Name
Return Type: string
Description: Given the name of a method, return a help string.
-----
Name : system.methodSignature( string )
Return Type: array
Description: Given the name of a method, return an array of legal signatures.
Each signature is an array of strings. The first item of each signature is
the return type, and any others items are parameter types.
-----
       : system.multicall( array )
Name
Return Type: array
Description: Process an array of calls, and return an array of results. Calls
should be structs of the form {'methodName': string, 'params': array}. Each
result will either be a single-item array containg the result value, or a
```

```
struct of the form {'faultCode': int, 'faultString': string}. This is useful
when you need to make lots of small calls without lots of round trips.
Name : system.shutdown( string )
Return Type: int
Description: Shut down the server. Return code is always zero.
```

# 4.3 XML RPC client examples

# 4.3.1 Minimal python example

XML RPC is very easy and pleasant to use in python.

For example to call method receiver.setCalibration on the server, you need only the following lines.

```
import xmlrpclib
server = xmlrpclib.ServerProxy("http://localhost:1080" )
print server.receiver.setCalibration( "sky" )
```

# 4.3.2 Python examples

See directory python/xmlrpc for other XML-RPC python examples.

# 4.3.3 C++ client example

hemt-cpp-GetStatus is a small example for XML-RPC, written in C++. The source code is available in apps/GetStatus

#### 4.4 hemt-init-ampli

This program initializes the HEMT amplifiers. It writes also a magic value in the amplifier volatile memory (ByteMemory=0xb6), to detect if the amplifiers are initialized or not.

#### 4.4.1 Syntax

```
$ hemt-init-ampli -h
HEMT Init Ampli - Init Ampli
Usage: hemt-init-ampli [options]
Options:
    -v Display version information
    -h, -? Display help
```

#### 4.5 hemt-check-ampli

This program checks that the amplifiers are initialized.

# 4.5.1 Syntax

```
$ hemt-check-ampli -h
HEMT Check Ampli - Check Ampli Status
Usage: hemt-check-ampli [options]
Options:
    -v Display version information
    -h, -? Display help
```

# 4.6 Simulator

The goal of this program is to simulate the HEMT receiver, so that the other software can be written before the receiver hardware is ready.

#### Syntax:

hemt-simul

# 4.6.1 Main window



The main window has three buttons, click on them to display/hide simulation window for LO, Mixer or Cryo.

The simulation occurs only when the associated subwindow is opened.

For example, if you want to simulate only the LO, open only the LO window.

# 4.6.2 LO Simulation Window

The LO window displays the simulator for the LO.

🔓 HEMT LO Simul	
File Maintenance	
[hemt] LO	
Motors	DACs
LoFreq 4.041 V R= 4.041 V	
Error Code 0 👻 Status off 👻	GunnBias 2.294 V
PwrGunn 1.304 V R= 1.304 V	Error Code 0
Error Code 🛛 📥 Status off 🛓	
HMxPower 1.851 V R= 1.851 V	LoopGain 2.294 V
Error Code 0 🔦 Status off 🛓	Error Code 0
LoPower1 2.400 V R= 2.400 V	
Error Code 0 🛉 Status off 🛓	HMx Bias 1.468 V
LoPower2 2.034 V R= 2.034 V	
Error Code 0 👻 Status off 👻	Error Code 0
Switches	ADCs
	Offset Voltage 3.125 V
aunn I ON OFF	Error Code 🛛 📥
loop   Close  Open	PLL IF Level
sweep 💿 ON 🔿 OFF	Error Code 🛛 📥
deltaF ○ +	HarmMxCurrent
Error Code 0	Error Code 0

# Motor simulator:

It simulates a CAN motor. It moves when position requests arrive. You can simulate the following failures:

- The motor returns an error code: enter the error code in the Error Code spinbox.
- The motor is missing: right-click, and unselect *"Enable"*
- The motor answers, but does not move: right-click and unselect "Auto"



When "Auto" is unselected, a !! symbol appears.

LoFreq	0.020 V !!	R= 0.000 V
Error Coo	de 0	

# Adc simulator

It simulates a CAN ADC. You can simulate a device missing error: right-click and unselect *"Enable"* 

# Dac simulator

It simulates a CAN DAC. You can simulate a device missing error: right-click and unselect *"Enable"* 

#### Lo Switches

It simulates the Lo switches.

You can simulate the following errors:

A device is missing error: right-click and unselect "Enable"

The device answers, but values do not switch: right-click and unselect "Auto"



When "Auto" is unselected, a warning appears in the widget: "Automatic Mode Disabled"



# 4.6.3 Ampli Simulation Window

This window simulates the amplifiers.

👔 HEMT Ampli S	imul				
File Maintenand	ce				
[hemt] Amp ampli	li				
Init value = 0 2010-08-21T12:J	15:02				
PowerAll value 2010-08-21T12:J	= 1 15:02				
ProtectionAll v	/alue = 255				
ByteMemory = 0>	(b6				
ampli_V1			ampli_V2		
	channels			channels	
Power = 1	ID = 27.75	Error Code 0	Power = 1	ID = 0.10	Error Code 🛛 📥
	VD = 0.20	Error Code 0		VD = 0.20	Error Code 0
Protection = 0	VG1 = 0.30	Error Code 0	Protection = 0	VGl = 0.30	Error Code 0
	VG2 = 0.40	Error Code 🛛 📥		VG2 = 0.40	Error Code 🛛 📥
ampli_H1			ampli_H2		
	channels			channels	
Power = 1	ID = 0.10	Error Code 0	Power = 1	ID = 0.10	Error Code 0
	VD = 0.20	Error Code 0		VD = 0.20	Error Code 0
Protection = 0	VG1 = 0.30	Error Code 0	Protection = 0	VG1 = 0.30	Error Code 0
	VG2 = 0.40	Error Code 0		VG2 = 0.40 ▲	Error Code 0

# 4.6.4 Cryo Simulation Window

🔓 Cryo Simul						
File Maintenance						
Temperatures	Lo2	Calibration	Attenuators			
Hot Load 13.961 degC	Command Running: 0 Status ✓ Locked ✓ Sts Running □ Cmd0	Load Command: Off Status Off ♀ Mirror Command: Off	Att. V 3.0 dB Error Code 0			
Cryostat Error Code 0	×	Status Off 🗢	Att. H 4.5 dB			
Cryo77K 48.536 K 15K P 15K Load 30.687 K Amp	32.256 K 32.716 K	Command: Off Status Off ¢	Error Code 0			

This window simulates:

- Hot load temperature
- Cryostat temperatures
- Calibration system

• Attenuators

You can simulate the following failures:

- The sensor returns an error code: enter the error code in the Error Code spinbox.
- The sensor is missing: right-click, and unselect "Enable"

#### Hot Load

Hot Load 0.000 degC					
	Hot Load Adc				
	√ Enable				
Ennon	Codo 0				

#### Cryostat

This widget simulates the different cryostat temperature. You can simulate the following failure:

• the sensor is missing: right-click, and unselect "Enable"

# **Calibration system**

This widget simulates the different cryostat temperature.

You can simulate the following failure:

• The calibration system is missing: right-click, and unselect "Enable"

#### Attenuators

This widget simulates the attenuators.

You can simulate the following failures:

- The attenuators return an error code: enter the error code in the Error Code spinbox.
- The attenuators are missing: right-click, and unselect "Enable"

#### 5 User Programs

This section describes the user programs. They are installed in /home/introot/hemt/bin.

There are also debug versions of these programs (.Debug suffix) The data files are installed in /home/introot/hemt/data

#### 5.1 hemt-i2c-reset

hemt-i2c-reset sends a CAN message to reset the CanI2c board.

#### 5.1.1 Syntax

```
$ hemt-i2c-reset -h
hemt-i2c-reset - Reset I2C bus
Usage: hemt-i2c-reset [options]
Options:
   -v Display version information
   -h, -? Display help
```

# 5.1.2 Example

```
$ hemt-i2c-reset
Reset command sent
```

# 5.2 UtilCan

For a complete description, see document *can-ip.pdf* 

# 5.2.1 Syntax

```
$ UtilCan -h
UtilCan - CAN Utility to read/write CAN messages
Usage: UtilCan [options]
Options:
   -p=N UDP port to contact (mandatory)
   -f=filename File to load (optional)
   -v Display version information
   -h, -? Display help
Example:
   UtilCan -p=2500 -f=myfile.txt
```

# 5.2.2 Screenshot

VilCan@localhost:2500 - Untitled (modified)	_ <b>_ X</b>
<u>F</u> ile Edit Help	
i 🔄 🔛 🔡 💦	
Can Message	
type here your new message	
<pre># comment starts with '#' # an extended message starts with X</pre>	Insert
X 0x00000001 [0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 ] # a standard message starts with S	Delete
S 0x00000002 [0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 ]	Replace
	Move Up
	Move Down
Send Msg one time	Exit

# 5.3 CanLogger

It is a CAN monitor tool, that can optionally decode the CanID into symbolic names. For a complete description, see document can-ip.pdf

# 5.3.1 Syntax

# \$ CanLogger -h

```
CanLogger - CAN logger

Usage: CanLogger [options]

Options:

-p=N UDP port to contact (mandatory)

-v Display version information

-h, -? Display help

Example:

CanLogger -p=2500

Note: If the environment variable CAN_DB_FILE is set,

the program loads the database to decode CanID into symbolic names.
```

# 5.3.2 Example

```
$ CanLogger -p=2501
Settings:
Port= 2501
DatabaseFile= /home/introot/hemt/data/hemt.db
Load data from /home/introot/hemt/data/hemt.db
2010-04-16T16:24:26.307: msg 1 : ampli_getByteMemory : X 0x010c0210 []
2010-04-16T16:24:26.307: msg 2 : ampli_H1_VD_get : X 0x010c0291 []
2010-04-16T16:24:26.307: msg 3 : ampli_H2_ID_get : X 0x010c0294 []
2010-04-16T16:24:26.307: msg 4 : ampli_H2_VD_get : X 0x010c0295 []
```

5.4 Lo

This program tunes the local oscillator.

#### 5.4.1 Syntax

```
HEMT Lo - Tune Local Oscillator

Usage: hemt-lo [options]

Options:

-f=frequency Specify the LO frequency in GHz

-d=[-|+] Specify the deltaF, '+' or '-'. Default is '-'.

-v Display version information

-h, -? Display help

Example:
```

hemt-lo -f=70.0

# 5.4.2 Example

```
$ hemt-lo -f=70
Setting for this tuning:
        bandNum = 2
        flo = 70.000 GHz
        deltaF = MINUS
FGUNN= 70 GHz
```

```
Load settings from h276.hemt
Step 1: Configure the LO in safe mode
Step 2: Set Motors and DACs
LO settings:
        AttH = 4.5
        AttV = 4
        FGunn = 70
        GunnBias = 7.8
        HarmMixerBias = 0.1
        HarmMixerPower = 0.4
        LoFreq = 5.542
        LoPower1 = 6.5
        LoPower2 = 9
        LoopGain = 4.747
        PowerGunn = 4.1
Step 3: Close Loop and optimize LoFreq
Optimal loFreq = 5.521
Step 4: Set Offset Voltage and Gunn Bias
Increase loFreq until OffsetVoltage < 5.00</pre>
loFreq = 5.5210 ; offsetVoltage = 4.0625
Increase gunnBias until OffsetVoltage > 5.00
Optimal gunnBias = 7.80968
Step 5: Find max PllIfLevel(harmMixerBias)
Max found: harmMixerBias = 1.1, pllIfLevel = 9.99985
Tuning Ok
Step 6: Set the attenuators
End of tuning
```

# 5.5 Gui

This program is specially designed for the front-end laboratory. It provides a graphical user interface (gui) for each receiver component.

#### Syntax

hemt-gui

#### 5.5.1 Main window

🐣 HEMT Gui [localhost]			
<u>F</u> ile	M <u>a</u> intenance	<u>H</u> elp	
		<u>A</u> mpli Exit	<u>C</u> ryo

# 5.5.2 Lo window

🐣 немт	LO Tuning			
File Mai	ntenance			
[hemt] Motors	LO			DACs
LoFreq	4.041 V	R= 4.041 V	Stop	
	();	:D	Reset	GunnBias 2.294 V R= 2.294 V
PwrGunn	1.304 V	R= 1.304 V	Stop	
			Reset	
HMxPower	1.851 V	R= 1.851 V	Stop	2.294 V R= 2.294 V
			Reset	
LoPowerl	2.400 V	R= 2.400 V	Stop	
			Reset	
LoPower2	2.034 V	R= 2.034 V	Stop	HMx Bias
			Reset	
Switches				ADCs
s	gunn	💿 ON	O OFF	Offset Voltage = 3.125 V
s	loop	🗿 Close	🔘 Open	PLL TE Level = 1.956 V
S	sweep	💿 ON	O OFF	
S	deltaF	0 +	0	HarmMxCurrent = 4.102 mA

# Motor widgets (LoFreq, PwGunn, HmxPwr, LoPower1, LoPower2)

This widget is used to drive the motor, and to display the current position.

# Description

– Motors –			
LoFreq	1.345 V	R= 8.022 \	/ Stop
L	-[	0	Reset

The requested value is displayed with a R prefix (here: R= 8.022).

The current position with no prefix (here: 1.345). There is also a stop button and a reset button.

To modify the motor position , you can:

•Click on the button name to open a dialog box to enter the new motor value •Move the slider

🐣 Enter LoFreq	? 🗆 🗙
Enter value for LoFreq:	
<u></u> K	Cancel

Motor dialog box

You can change the slider step by right-clicking on it

# **HEMT Software**

2.068 V R= 2.068 V	🐣 LoFreq: set slider page step 🛛 ? 🗆 🗙
	Enter new slider step for LoFreq: (current step is 10)
Pw Gupp 0 Current step = 10	
Right-click to modify	<u>O</u> K Cancel
UMY Dur 0.020 V R= 0.020 V	

The unit for the slider is in motor raw unit (totally different from the motor unit which is displayed in Volt)



If the motor is disconnected from the CAN bus, the labels become red to indicate a problem.

# DAC widget (GunnBias, LoopGain, HarmMxBias)



This widget is used to drive the DAC, and to display the current position.

#### Description

From the user point of view, a DAC is similar to a motor with an infinite speed.

So, the motor widget description applies also to the DAC widget.

# ADC widget (OffsetVoltage, PLL IF Level, HarmMxCurrent)

ADCS OffsetVoltage (V) = 0.000 PLL IF Level (V) = 0.000 Harm Mx Current (mA) = 0.000

This widget is used to display the current ADC value.

#### Lo Switches widgets

This widget is used to command the LO switches and to display the status.

#### Description

Each radio-button pair represents a LO switch. It is a realistic representation of the LO physical front panel: color and switch order are taken from the LO hardware.

– Switche	es		
S	gunn	• ON	OFF
S	loop	Close	● Open
S	sweep	• ON	OFF
S	deltaF	0 +	• -

The radio button displays the current command applied on this switch.

For each switch, if the command matches the status, the status label (the S letter on the left side) is displayed with a normal background; otherwise this label is displayed with a red background. Here the status for Sweep is red. It means that the status does not match the command.

- Switches			
S	gunn	● ON	OFF
S	loop	Close	🔵 Open
S	sweep	● ON	OFF
S	deltaF	• +	0 -

If the LoSwitches device does not answer at all, the switch names (gunn, loop, sweep, deltaF) become red to indicate a problem.

## 5.5.3 Amplifiers window

🐣 HEMT Ampli Tuning 📃 🗆 🗙		
File Maintenance		
[hemt] Ampli	i	
Ampli Ampli are initiali	zed	
Ampli Vl	Ampli V2	
Power On 🖨	Power On 🖨	
Protection Off 🖨	Protection Off 🖨	
ID = -0.17	ID = -27.93	
VD = -1.30	VD = -1.30	
VG1 = -1.20	VG1 = -1.20	
VG2 = -1.10	VGZ = -1.10	
Ampli Hl	Ampli H2	
Power On 🖨	Power On 🖨	
Protection Off 🖨	Protection Off 🖨	
ID = -27.85	ID = -28.01	
VD = -1.30	VD = -1.30	
VG1 = -1.20	VG1 = -1.20	
VG2 = -1.10	VG2 = -1.10	

This window displays the amplifiers currents and voltages.

#### Amplifiers initialization

If the amplifiers are not initialized, a warning message and a button *Init* appear. Click on the button to initialize the amplifiers.



# Summary

The amplifiers work only when all the following conditions are satisfied.

- The amplifiers are initialized
- the physical switch (on the amplifier box) on the HEMT receiver is on the *Unprotected* position.
- the software power is ON
- the software protection is OFF

# 5.5.4 Cryo window

🐣 Cryo GUI	
File Maintenance	
Temperatures	L02
Hot Load = 13.961 degC	Command
- CryoStat	Running: On 😫
Cryo77K: 48.536 K 15K Plate: 32.256 K	Status Locked: L
15K E080. 50.007 K Amptil. 52.710 K	Running: I Cmd Bito, O
Calibration	
Position Load: Off ♦ Mirror: Off ♦	Table: Off 🖨
off 🗢 Status: Off Status: Off	Status: Off
Warm IF	
Att. V 3.0 dB	. H <mark>4.5</mark> dB 🔶

This window displays

- the cryostat temperatures
- the calibration commands
- the Lo2 commands
- the attenuators commands

L02		If the
Command		
Running:	On 🗢	
Status		
Locked:	0	
Running:	0	
Cmd Bit0:	0	

the Lo2 is unlocked, the widget becomes orange.

# 5.6 Rop

Rop (Receiver OPerator) is a special version of hemt-gui. All components are grouped on only one window,

The target audience for this program is the IRAM 30M telescope operators.

😞 HEMT ROP [localhost] <@mrt-hemt>	
<u>F</u> ile <u>M</u> aintenance <u>H</u> elp	
[mrt-hemt] LO	[mrt-hemt] Ampli
- Motors	_ Ampli
LoFreq 3.738 V R= 3.738 V GunnBias 7.929 V R= 7.929 V	Ampli are initialized
PwrGunn 1.418 V R= 1.416 V	Ampli V1
HMxPower 0.305 V R= 0.305 V LoopGain 4.584 V R= 4.584 V	Protection Off ▼ ID = 4.03 ID = 10.86
LoPover1 7.502 V R= 7.505 V	VD = 0.97 VD = 1.07
LOI 0 WEIT	VG1 = 0.16 VG1 = 0.22
LoPower2	VG2 = 0.16 VG2 = 0.24
- Switches	_ Ampli H1 Ampli H2
	Protection Off 👻 Protection Off 👻
G lass O Glass O Grass	ID = 7.03 ID = 10.21
PLL IF Level = 1.599 V	VD = 1.00 VD = 0.80
S sweep ON OFF	VG1 = 0.17 VG1 = 0.18
S deltaF + O - HarmMxCurrent = 1.377 mA	VG2 = 0.21 VG2 = 0.21
- Temperatures CryoStat	
Hot Load = 25 188 degC Cryo77K: 64.257 K 15K Plat	te: 12.376 K
15K Load: 12.528 K Ampli	i: 12.678 K
- Calibration	Lo2
Calibration	Command
Position Load: Off V Mirror: On V Tab	le: Off 🔻 Running: On 💌
Status: Off Status: On Stat	us: Off
sky	Status
- Warm IF	Locked:
Att. V 2.5 dB 🚔 Att. H 2.5 d	B Cmd Bit0: 0

See hemt-gui section for hemt-rop usage instructions.

# 5.7 hemt-GetStatus

hemt-GetStatus.py is a small program that send a receiver.getStatus() to the hemt-server, and then print the result on the standard output.

# 5.7.1 Syntax

<pre>\$ hemt-GetStatus.py</pre>	-h	
Print receiver statu	S	
Syntax:		
hemt-GetStatus.p	y options	
Options		
-s=serverName	Default=localhost	
-h, -?	Print help	

# 5.7.2 Example

<pre>\$ hemt-GetStatus.py</pre>				
# Connect to http://localhost:1080				
status.attenuatorH	0.0			
status.attenuatorV	0.0			
status.deltaf	0			
status.gunn	1			

status.loop	0
status.sweep	0
status.temperature.Amp15K	11.952319
status.temperature.Band1	12.714197
status.temperature.Band3	12.619856
status.temperature.Band4	12.333979
status.temperature.ColdLoad	35.994822
status.temperature.Cryo15K	35.972908
status.temperature.Cryo4K	12.79326
status.temperature.Cryo77K	48.536285
status.temperature.hotLoad13	273.1
status.temperature.hotLoad24	273.1

# 5.8 Check software

hemt-check-software.py is a small program to check if all the HEMT programs run normally:

#### 5.8.1 Syntax

```
$ hemt-check-software.py -h
hemt-check-software.py - Check hemt software status
Usage: hemt-check-software.py [options]
Options:
    -h,-? Print this help
    -v Print version
```

Run hemt-check-software.py without options to start checkings.

#### 5.8.2 Example

HEMT Receiver: Ok

# 5.9 hemt-SetProtection.py

hemt-SetProtection.py is a command line program to set/unset the amplifiers protection.

```
5.9.1 Syntax
```

#### 5.9.2 Example

```
$ hemt-SetProtection.py --amplifier=all --protection=0
connect to http://localhost:1080
ampli = all
protection = 0
result = 0
```

# 5.9.3 Alternative methods

There are two other ways to set/unset the amplifiers protection:

- by using the graphical program (hemt-rop or hemt-gui)
- by using the XML-RPC procedure ampli.setProtection

# 6 Daily Operation

# 6.1 Modify database configuration

The configuration settings are stored in several SQL files in directory /home/introot/hemt/data/. Normals users are interesting only by the filenames starting with prefix "conf-". The others filenames are for developers only.

To modify a setting, edit the appropriate file and then run hemt-update-db.sh to update the database.

```
$ hemt-update-db.sh
    cd /home/introot/hemt/data
    rm -f hemt.db
# Update hemt.db ...
    sqlite3 hemt.db < ./00-canid.sql
    sqlite3 hemt.db < ./conf-cryo.sql
    sqlite3 hemt.db < ./conf-lo.sql
    sqlite3 hemt.db < ./converters/LinearConverter.sql
    sqlite3 hemt.db < ./converters/RawConverter.sql
    sqlite3 hemt.db < ./devices/Amplis.sql
    sqlite3 hemt.db < ./devices/CanAdc.sql
    sqlite3 hemt.db < ./devices/CanButton.sql
    sqlite3 hemt.db < ./devices/CanButton.sql
    sqlite3 hemt.db < ./devices/CanDac.sql
    sqlite3 hemt.db < ./devices/CanMotor.sql</pre>
```

```
sqlite3 hemt.db < ./devices/CanRegister.sql
# OK</pre>
```

Then you have to restart the applications.

Note: You should notify me by email each time you modify SQL file, so that I can archive it in the SVN repository. Otherwise your modifications may be lost after a reinstallation.

# 6.2 LO data files

The LO data files are in /home/introot/hemt/data/tuning If you add a new file in this directory, you have to specify its names in the appropriate SQL configuration file (/home/introot/hemt/data/conf-\*.sql).

# 7 Tips

# 7.1 How to change the fonts size ?

The graphical program use the default setting from your window manager, but you can change the default font with the program qtconfig-qt4

🔌 Qt Configuration <@mi	🔌 Qt Configuration <@mrt-emir1> 📃 🗆 🗙					
<u>F</u> ile <u>H</u> elp						
Fonts		Appearance Fonts Interface Printer Phonon				
		Default Font				
Use this tab to select		F <u>a</u> mily: Bitstr	eam Vera Sans Mono 🕞 🗸			
the default font for		Style:				
The selected font is		Point Size:				
shown (initially as						
'Sample Text') in the						
Family. Style and		Font Substitution				
Point Size drop down	ze drop down Select or Enter a Family: Bitstream Charter		eam Charter			
lists.						
Qt has a powerful font substitution feature that allows you to specify a list of substitute fonts. Substitute fonts are used when a font cannot be loaded, or if the specified font doesn't have a particular character.						
For example, if you		Up Down	Remove			
select the font Lucida, which doesn't have Korean	•	Select substitute Family: Bitstream Charter	Add			

To have a nicer display, you should select a fixed-width font. I recommends to use *Bitstream Vera Sans Mono*, with size=10 or size=12. (on Debian this font is in the ttf-bitstream-vera package)

# 8 Troubleshooting

# 8.1 IPv6

It seems that CanManager does not work when IPv6 is enabled. On debian you can disable IPv6 with the following commands:

```
# echo net.ipv6.conf.all.disable_ipv6=1 > /etc/sysctl.d/disableipv6.conf
# /sbin/reboot
```

# 8.2 Procedure to reset the I2C bus

If the hemt-rop window looks like (red colors can be continuous or blicking)

💑 HEMT ROP [localhost]	_ 🗆 🗡
<u>F</u> ile <u>M</u> aintenance <u>H</u> elp	
[hemt] LO	[hemt] Ampli
Motors DACs	Ampli
LoFreq 0.027 V R= 5.520 V	Ampli are NOT initialized
PwrGunn 0.027 V R= 4.099 V	Ampli V1 Ampli V2
HMxPower 0.027 V R= 0.398 V LoopGain 0.000 V R= 4.747 V	Protection Off ID = 0.16 ID = 0.13
LePeyer] 0.027 V R= 6.499 V	VD = 0.00 $VD = 0.00$
0.000 V R= 1.100 V	VG1 = -0.00 $VG1 = -0.00VG2 = 0.00$ $VG2 = 0.00$
LoPower2	Ampli H1
Switches ADCs	Protection Off
S gunn 💿 ON 📀 OFF Offset Voltage = 0.000 V	
S loop Close Open	VD = 0.00 $VD = 0.00$
S sweep ON OFF	VG1 = -0.00 VG1 = -0.00
S deltaF 🔾 + 💽 - HarmMxCurrent = 0.000 mA	VG2 = 0.00 $VG2 = 0.00$
Temperatures	
Hot Load = 0.000 degC Cryo77K: 48.536 K 15K Plat	te: 32,256 K
- Calibration	Lo2
Calibratian	Command
Position Load: Off  Mirror: Off  Table	.e: Off 🗢 Running: On
status: Off Status: Off Status:	is: Off
	Status
Warm IF	Locked: 0
Att. V 0.0 dB 🔶 Att. H 0.0 d	IB Cmd Bit0: 0

It means that the I2C bus has crashed, i.e. a I2C module hangs the I2C bus for ever. Consequence: the receiver is still running, but you cannot send any new command to the I2C modules until you reset the I2C. bus

Procedure to reset the I2C bus

- Since it is impossible to reset the I2C bus remotely, you have to climb up into the cabin.
- Go to the receiver back
- On the box *HEMT 3mm BIAS* ( a black box on the top of the cryostat), and switch the protection switch to *Protected* (up position)
- Go back to the receiver front
- Switch OFF the CANI2C rack (square button on the top)
- Wait 5 seconds
- Switch ON the CANI2c rack
- Go to the receiver back
- On the *HEMT 3mm BIAS* box, switch the protection switch to *Unprotected* (down position)