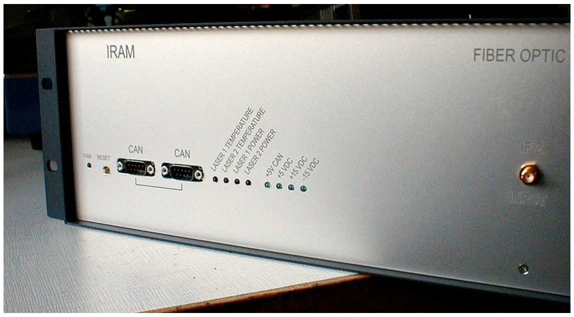
IRAM-COMP-068

Revision: 1
2010-03-30

Contact Author

**Institut de RadioAstronomie Millimétrique**

# Fiber Optic Software



Owner Sebastien Blanchet

**Keywords: fiber optic, software**

Approved by:
A.Perrigouard

Date:
March 2010

Signature:

# *Change Record*

| REVISION | DATE | AUTHOR | SECTION/PAGE AFFECTED | REMARKS |
|---|---|---|---|---|
| 0 | 2006-09-15 | S.BLANCHET | See IRAM-COMP-034 | Legacy document |

# Content

## 1 Introduction

In October 2006, IRAM has installed a new fiber optic transmission system to connect the new receivers (in each antenna cabin) to the correlator (in the computer room). This document is the reference for fiber optic software: installation, technical documentation, daily usage and troubleshooting.

You need also the hardware documentation of the Fiber Optic Processor from Philippe Chavatte: *FO_TX.pdf* and *FO_RX.pdf*

## 2 Presentation

2 devices compose the fiber optic transmission system:

- An optical transceiver, named FO_TX
- An optical receiver, named FO_RX

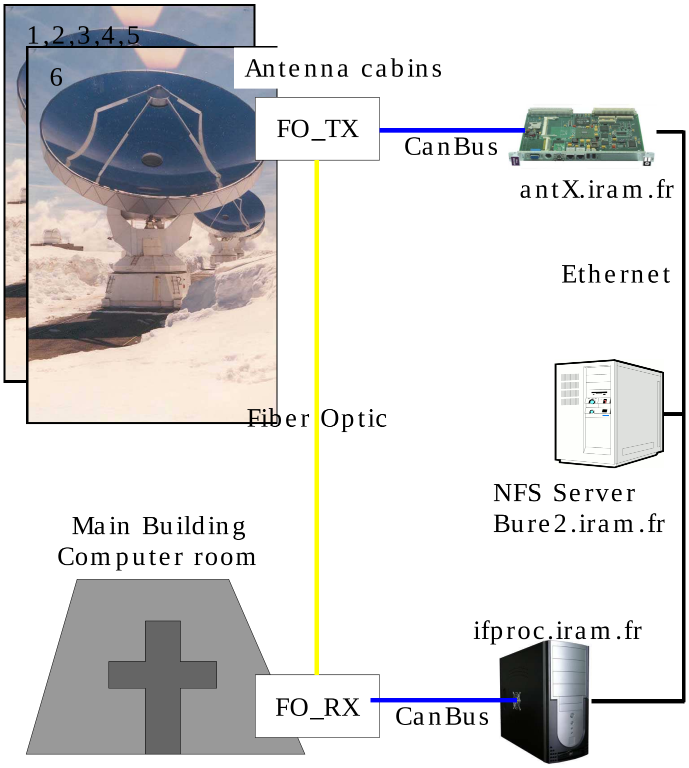


*Figure 1 System overview*

### 2.1 FO_TX

There are six FO_TX, one per antenna. They are located in the antenna cabin. These devices are autonomous, but to allow remote maintenance, a monitoring software can be run from antX.iram.fr

### 2.2 FO_RX

There is only one FO_RX. It is located in the computer room. All the fiber are connected to FO_RX. This device is driven by a computer through a CAN bus. The controlling software runs on *ifproc.iram.fr*.

*Ifproc.iram.fr* runs also the IF Processor software. (see the *IF Processor Software* documentation)

# 3 Installation

## 3.1 Requirements

### 3.1.1 Hardware

The minimal requirements for hardware are:

1. CPU: x86 CPU at 500 MHz
2. RAM: 512 MB
3. CAN controller: TPMC816 PMC card from Tews Technologies GmbH

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

### 3.1.2 Software

**Operating system**

The software should run on any recent Linux (i386 or x86_64) with a 2.6.x kernel.

**Mandatory software**

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 control system. | >= 1.5 | http://subversion.tigris.org |
| Qt | C++ Cross-platform application framework | >= 4.4.3 | http://www.qtsoftware.com |
| Sqlite | Embedded SQL database | >= 3.3.6 | http://sqlite.org |
| CxxTest | Unary testing framework for C++ | >= 3.10.1 | http://cxxtest.tigris.org |
| Xmlrpc-c | XML-RPC for C and C++. | >= 1.06 | http://xmlrpc-c.sourceforge.net |

All these libraries are very common (except CxxTest), 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 | C and C++ Integrated Development Environment (IDE) for the Eclipse platform. | >= 3.5 | http://eclipse.org/cdt |
| doxygen | Automatic documentation system | >= 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.

### 3.1.3 Network

For the real exploitation, the Fiber Optic software will run on a disk-less computer. So an NFS server is required to export the filesystem.

## 3.2 Building

### 3.2.1 Extract code from the repository

```
$ mkdir ~/develSVN
$ cd ~/develSVN
$ svn co svn://svn.iram.fr/PdB/FiberOptic/trunk FiberOptic
```

Then use the Makefile to extract automatically the dependencies

```
$ cd FiberOptic
$ qmake
$ make get_deps
```

### 3.2.2 Build the TPMC816 driver

The TPMC816 driver is optional for the simulation. In this case, you can skip this section.

```
$ cd ~/develSVN/tpmc816
$ su -c "make install"
```

To create devices (and to load the driver)

```
$ su -c "./create_devices.sh"
```

The `create_devices.sh` script creates nodes in `/etc/udev/devices` .

### 3.2.3 Build the FiberOptic code

```
$ cd ~/develSVN/FiberOptic
$ ./build.sh
```

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

```
$ make doc
```

To install the software its default settings in `/home/introot/fo`
Note: the shared libraries are installed in `/home/introot/lib`

```
$ su -c "./install.sh"
```

**Warning:**
Normally, you need not to change the default settings, but if you wish to update only the software **without reinstalling the default settings**, use

```
$ make -f Makefile.install programs
```

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

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

## 3.3 Configure environment

You should add `/home/introot/ifproc/bin` to your path.

If you wish to use CanLogger, you should also define `CAN_DB_FILE` as follow:

```
export DEVICE_NAME=fo
export CAN_DB_FILE=/home/introot/${DEVICE_NAME}/data/${DEVICE_NAME}.db
```

### 3.4 Initial tests

For this initial test, we run

1. The CanManagers to enable the Can Over IP protocol
2. The fo-rx simulator
3. The python xml-rpc client: `GetStatus.py`

First we remove the `/dev/tpmc816_*` devices, so the CanManager will run in simulation mode

```
$ su –c "rm –f /dev/tpmc816_*"
$ fo-init-can.sh
$ xterm -e fo-rx-simul &
$ xterm -e fo-rx-server &
```

Now run the `GetStatus.py`

```
# Connect to http://localhost:1088
status.canErrors                  0
status.laserH.a1                  0.1
status.laserH.a2                  0.2
status.laserH.a3                  0.3
status.laserH.a4                  0.4
status.laserH.a5                  0.5
status.laserH.a6                  0.6
status.laserV.a1                  1.1
status.laserV.a2                  1.2
status.laserV.a3                  1.3
status.laserV.a4                  1.4
status.laserV.a5                  1.5
status.laserV.a6                  1.6
status.output                     0
status.powerSupply.can5V          5.0
status.powerSupply.digital12V     12.0
status.powerSupply.digital15V     15.0
status.powerSupply.switch5V       5.09
status.temperature                25.0
status.temperatureDateTime        20100330T08:41:43
# GetStatus() rpc call takes  0.00681900978088  seconds
```

### 3.5 Start application automatically

Add `/home/introot/fo/bin/fo-init-device.sh` to `/etc/rc.local` to initialize the Fiber Optic software at the startup.

This script starts the CanManager and the fo-rx-server.

## 4 Internal programs

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

### 4.1 CanManager

For a complete description see the document IRAM-COMP-057 *CanIp*

```
$ CanManager -h
```

```
CanManager is a bridge between the CAN bus and the CAN/IP protocol
Usage: CanManager [options]
Options:
  -d=name        CAN controller device name to use. If missing,
                      the application runs in simulation mode
  -p=N           Listen to UDP port N
  -l=N           Limit write speed base CAN messages.
                      'N' is in message/sec. Default value = 0 (no limit)

  -v             Display version information
  -h, -?         Display help


Example:
        CanManager -d=/dev/tpmc816_0 -p=2500 -l=20
```

For the FiberOptic software, CanManager must listen on port udp/2501

```
CanManager –d=/dev/tpmc816_1 –p=2501 –l=50
```

## 4.2 FO RX Server

FO RX server is a XML-RPC server to remotely control FO RX.

> **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. This protocol is very simple to use, and can be used from any programming language (many opensource libraries are available)
>
> For a detailed introduction to XML RPC see http://en.wikipedia.org/wiki/XML-RPC

fo-rx-server listens for XML-RPC calls on http://localhost:1088/RPC2

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 http://xmlrpc-c.sourceforge.net/introspection.html
- multicalls

### 4.2.1 Syntax

```
$ fo-rx-server -h
Fo Server - XML-RPC Server
Listen on port 1088
Usage: fo-rx-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

```
$ cd ~/build/FiberOptic/python/xmlrpc/
$ ./ListMethods.py
# Connect to http://localhost:1088
forx.getStatus
forx.getTemperature
forx.initIo
forx.reset
forx.selectOutput
system.listMethods
system.methodHelp
system.methodSignature
system.multicall
system.shutdown
```

```
$ ./Introspection.py
# Connect to http://localhost:1088
----------------------------------------------------------------------
Name       : forx.getStatus(  )
Return Type: struct
Description: Returns the device status
Syntax: getStatus()


----------------------------------------------------------------------
Name       : forx.getTemperature(  )
Return Type: struct
Description: Get FO Rx temperature
Syntax: getTemperature()
Return a struct
  {
    double   temperature // temperature in Celcius degree
    dateTime temperatureDateTime // Date time of the temperature reading
  }


----------------------------------------------------------------------
Name       : forx.initIo(  )
Return Type: int
Description: Init IO to their default values
Syntax: initIo()
Return code is always 0


----------------------------------------------------------------------
Name       : forx.reset(  )
Return Type: int
Description: reset FO Rx
Syntax: reset()
Return code is always 0


----------------------------------------------------------------------
Name       : forx.selectOutput( int )
Return Type: int
Description: Select Output for FO Rx
Syntax: selectOutput( int a_output )
  a_output:
      0 -> noise
      1 -> receiver
Return code is always 0


----------------------------------------------------------------------
Name       : system.listMethods(  )
Return Type: array
Description: Return an array of all available XML-RPC methods on this server.

----------------------------------------------------------------------
```

```
Name       : system.methodHelp( string )
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.

----------------------------------------------------------------------
Name       : system.multicall( array )
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 Python examples

#### 4.3.1.1 Minimal example

XML RPC is very easy to use in Python.

For example to call method `forx.selectOutput(1)` on the server, you need only the following lines.

```
import xmlrpclib
server = xmlrpclib.ServerProxy( "http://localhost:1088" )
print server.forx.selectOutput( 1 )
```

#### 4.3.1.2 Other Python examples

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

### 4.3.2 Fortran examples

Unfortunately, there are no native XML-RPC libraries for Fortran, therefore Fortran programs have to use the C library for XML-RPC. ( http://xmlrpc-c.sourceforge.net )
For convenience, I have written a C library to hide the XML-RPC. Therefore Fortran programs can call remote procedures with simple C calls.

```
C Header:           libs/Rpc/FoRx_Rpc_Client.hpp
```

```
Fortran types:      libs/Rpc/CF_FoRx_Types.f
Compiled library:   bin/release/libfoRpc.so
Fortran example:    apps/Fortran/TestRpc
```

Notes:

- All the Fortran calls are described in the header file. There is one C function per XML procedure.
- The Fortran type file shares data types between C and Fortran. It is generated with `c2f`.
- The library has a C interface, but it is written in C++/QT. Therefore QT is required when linking with `foRpc`.
- If you wish to redevelop your own Fortran wrapper for `xmlrpc-c`, it is worth looking into the library source (`libs/Rpc`), to have an example.

For example to call method `forx.selectOutput(1)`from a Fortran program.

```
PROGRAM test_forx_select_output
CHARACTER(64) serverName
INTEGER output

serverName = ′localhost′
ouput = 1

CALL frpc_forx_select_output( TRIM(serverName), 1 )
STOP
END
```

### 4.3.3 Client/server performances

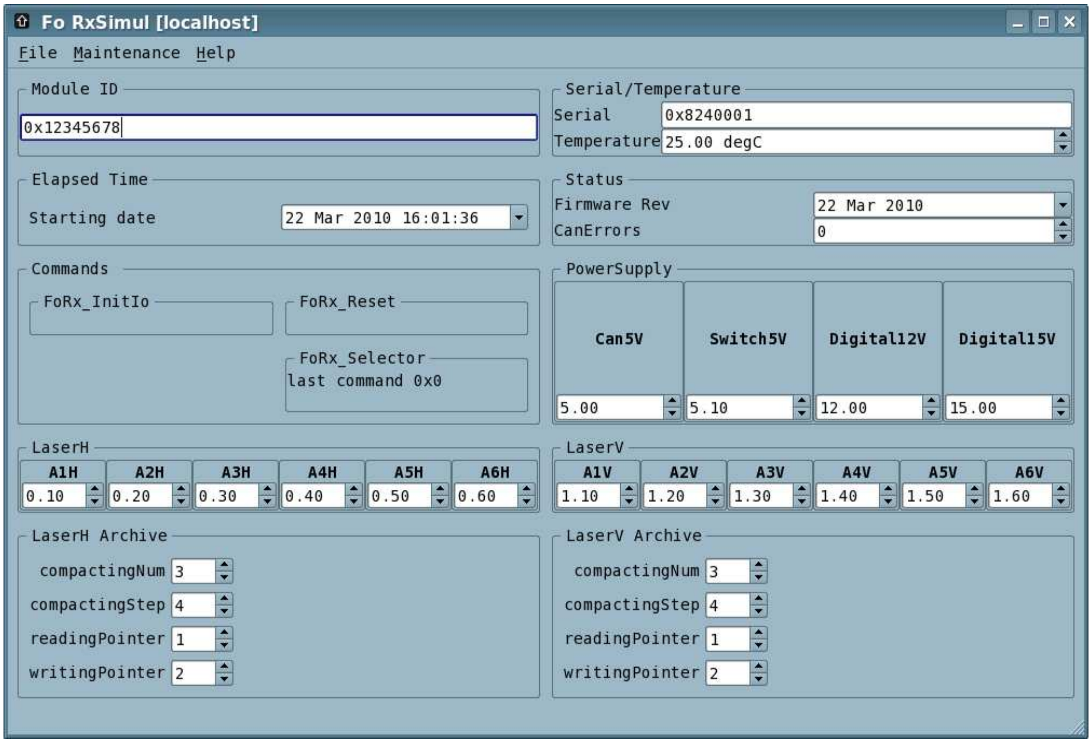
From the client point of view, the total call time (sending + request processing + status receiving) is slightly lower than 10 milliseconds. So we have very good performances.

## 4.4 FO RX Simulator

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

**Syntax:**

```
   fo-rx-simul
```



Module ID

0x12345678

Serial/Temperature

Serial 0x8240001

Temperature 25.00 degC

Elapsed Time

Starting date 22 Mar 2010 16:01:36

Status

Firmware Rev 22 Mar 2010

CanErrors 0

Commands

FoRx_InitIo

FoRx_Reset

FoRx_Selector

last command 0x0

PowerSupply

| Can5V | Switch5V | Digital12V | Digital15V |
|---|---|---|---|
| 5.00 | 5.10 | 12.00 | 15.00 |

LaserH

| A1H | A2H | A3H | A4H | A5H | A6H |
|---|---|---|---|---|---|
| 0.10 | 0.20 | 0.30 | 0.40 | 0.50 | 0.60 |

LaserV

| A1V | A2V | A3V | A4V | A5V | A6V |
|---|---|---|---|---|---|
| 1.10 | 1.20 | 1.30 | 1.40 | 1.50 | 1.60 |

LaserH Archive

compactingNum 3

compactingStep 4

readingPointer 1

writingPointer 2

LaserV Archive

compactingNum 3

compactingStep 4

readingPointer 1

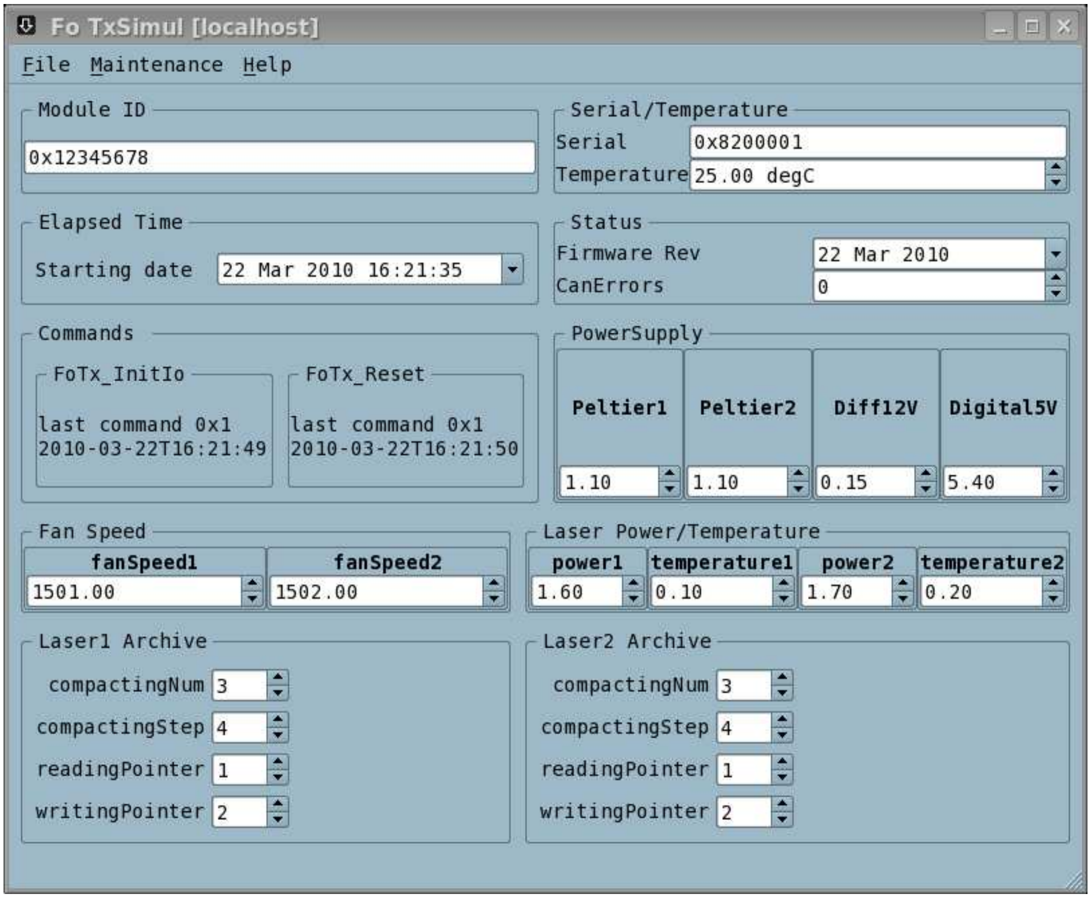
writingPointer 2

## 4.5 FO TX Simulator

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

**Syntax:**

```
fo-tx-simul
```



Module ID
0x12345678

Serial/Temperature
Serial 0x8200001
Temperature 25.00 degC

Elapsed Time
Starting date 22 Mar 2010 16:21:35

Status
Firmware Rev 22 Mar 2010
CanErrors 0

Commands
FoTx_InitIo
last command 0x1
2010-03-22T16:21:49
FoTx_Reset
last command 0x1
2010-03-22T16:21:50

PowerSupply

| Peltier1 | Peltier2 | Diff12V | Digital5V |
|---|---|---|---|
| 1.10 | 1.10 | 0.15 | 5.40 |

Fan Speed

| fanSpeed1 | fanSpeed2 |
|---|---|
| 1501.00 | 1502.00 |

Laser Power/Temperature

| power1 | temperature1 | power2 | temperature2 |
|---|---|---|---|
| 1.60 | 0.10 | 1.70 | 0.20 |

Laser1 Archive
compactingNum 3
compactingStep 4
readingPointer 1
writingPointer 2

Laser2 Archive
compactingNum 3
compactingStep 4
readingPointer 1
writingPointer 2

# 5 User programs

## 5.1 fo-rx-gui

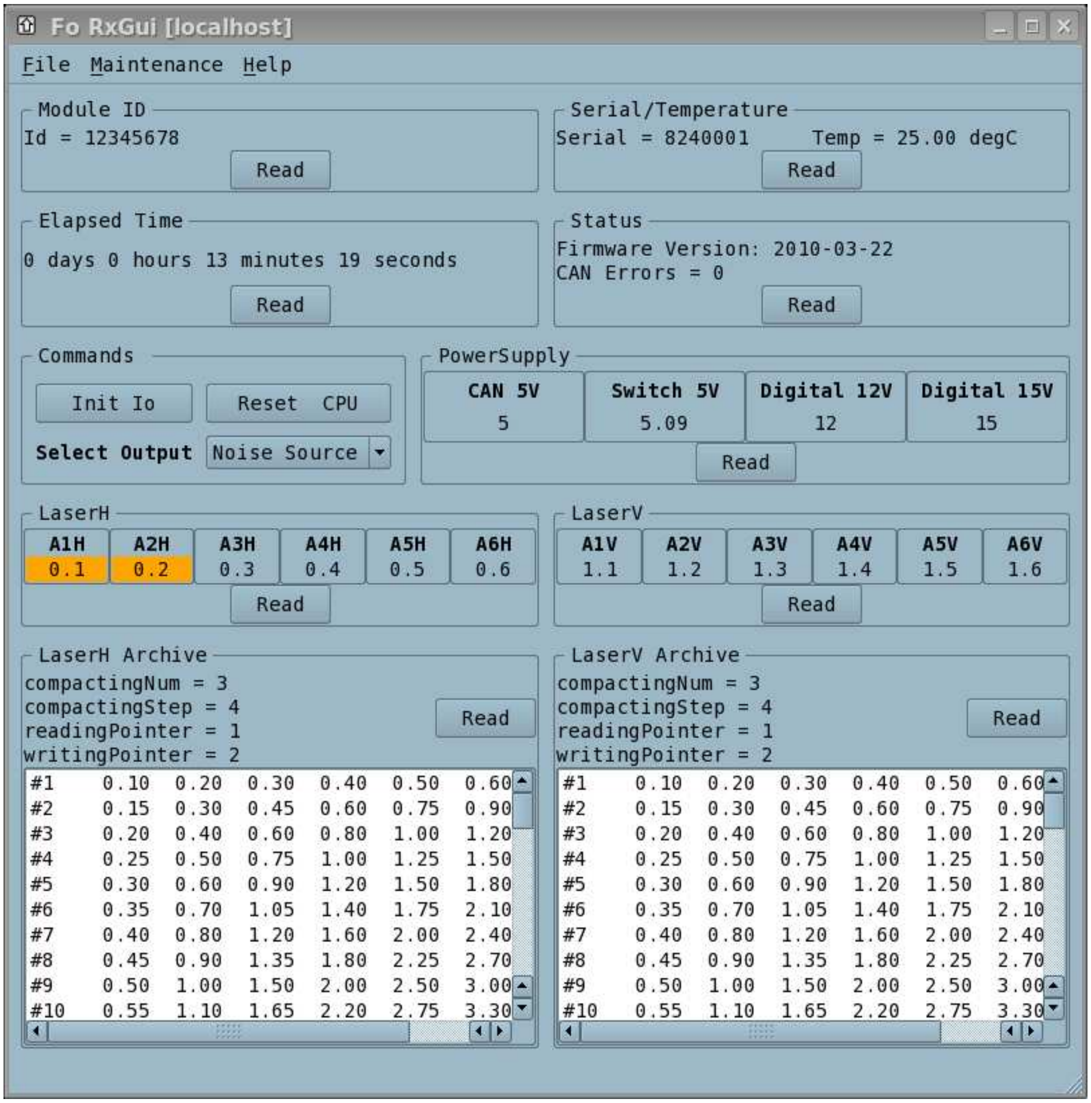
Fo-rx-gui is a graphical software to send command to the FO_RX device.

**Syntax:**

```
    fo-rx-gui
```

Procedure to log in forx.iram.fr, and execute the program

```
$ ssh –Y backend@forc.iram.fr
backend@forx.iram.fr password:
$ fo-rx-gui
```



Module ID
Id = 12345678

Serial/Temperature
Serial = 8240001 Temp = 25.00 degC

Elapsed Time
0 days 0 hours 13 minutes 19 seconds

Status
Firmware Version: 2010-03-22
CAN Errors = 0

Commands
Select Output: Noise Source

PowerSupply

| CAN 5V | Switch 5V | Digital 12V | Digital 15V |
|---|---|---|---|
| 5 | 5.09 | 12 | 15 |

LaserH

| A1H | A2H | A3H | A4H | A5H | A6H |
|---|---|---|---|---|---|
| 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 |

LaserV

| A1V | A2V | A3V | A4V | A5V | A6V |
|---|---|---|---|---|---|
| 1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 1.6 |

LaserH Archive

```
compactingNum = 3
compactingStep = 4
readingPointer = 1
writingPointer = 2
#1     0.10  0.20  0.30  0.40  0.50  0.60
#2     0.15  0.30  0.45  0.60  0.75  0.90
#3     0.20  0.40  0.60  0.80  1.00  1.20
#4     0.25  0.50  0.75  1.00  1.25  1.50
#5     0.30  0.60  0.90  1.20  1.50  1.80
#6     0.35  0.70  1.05  1.40  1.75  2.10
#7     0.40  0.80  1.20  1.60  2.00  2.40
#8     0.45  0.90  1.35  1.80  2.25  2.70
#9     0.50  1.00  1.50  2.00  2.50  3.00
#10    0.55  1.10  1.65  2.20  2.75  3.30
```

LaserV Archive

```
compactingNum = 3
compactingStep = 4
readingPointer = 1
writingPointer = 2
#1     0.10  0.20  0.30  0.40  0.50  0.60
#2     0.15  0.30  0.45  0.60  0.75  0.90
#3     0.20  0.40  0.60  0.80  1.00  1.20
#4     0.25  0.50  0.75  1.00  1.25  1.50
#5     0.30  0.60  0.90  1.20  1.50  1.80
#6     0.35  0.70  1.05  1.40  1.75  2.10
#7     0.40  0.80  1.20  1.60  2.00  2.40
#8     0.45  0.90  1.35  1.80  2.25  2.70
#9     0.50  1.00  1.50  2.00  2.50  3.00
#10    0.55  1.10  1.65  2.20  2.75  3.30
```

With this software, you can

- read all the FO_RX internal values. If the value is out of range, the widget background becomes orange.
- select the wanted output (Noise Source or Receiver)
- reset the FO_RX microcontroller
- download the Laser EEPROM data, which hold the history.

Note 1:
By default, the software runs in manual mode, i.e. you must click on the *Read* buttons to get the FO_RX values. Nevertheless, the automatic refreshing can be enabled by opening the menu *"Maintenance | Refresh"*

Note 2:
You have to click two times on Read, to display the Laser Records
The reason is simple: we cannot know before how many data will be sent by the device. It depends on the EEPROM content, so it is difficult to know when the widget should be updated.

## 5.2 fo-rx-report

Fo-rx-report is a command line program to generate a report about FO RX.

**Syntax**

```
$ fo-rx-report -h
Fo Rx Report - Print FoRx report
Usage: fo-rx-report [options]
Options:
  -v              Display version information
  -h, -?          Display help
```

Tip: Use redirection to save the program output in a filename.

```
$ fo-rx-report > myreport.txt
```

Example of report

```
############################################################
# Fiber Optic RX - Automatic Report
# Date: 2010-03-22T16:13:18
# Generated on: gre106
############################################################



############################################################
Status
moduleId = 12345678
serial = 136577025
temperature = 25 deg Celcius
Firmware Version = 2010-03-22
Can Error = 0
############################################################

############################################################
Selection
Output = 0

############################################################
Elapsed Time = 0 days 0 hours 11 minutes 41 seconds

############################################################
PowerSupply
Can 5V = 5
Switch 5V = 5.09
Digital 12V = 12
Digital 15V = 15

############################################################
LaserH
A1H = 0.1
A2H = 0.2
A3H = 0.3
A4H = 0.4
A5H = 0.5
A6H = 0.6

############################################################
LaserV
A1V = 1.1
A2V = 1.2
A3V = 1.3
A4V = 1.4
A5V = 1.5
A6V = 1.6
```

```
############################################################
LaserH Archive
compactingNum = 3
compactingStep = 4
readingPointer = 1
writingPointer = 2
#1    0.10  0.20  0.30  0.40  0.50  0.60
#2    0.15  0.30  0.45  0.60  0.75  0.90
#3    0.20  0.40  0.60  0.80  1.00  1.20
#4    0.25  0.50  0.75  1.00  1.25  1.50
[...]

# End of Report
```

### 5.3 fo-tx-gui

Fo-tx-gui is a graphical software to send command to the FO_TX device.

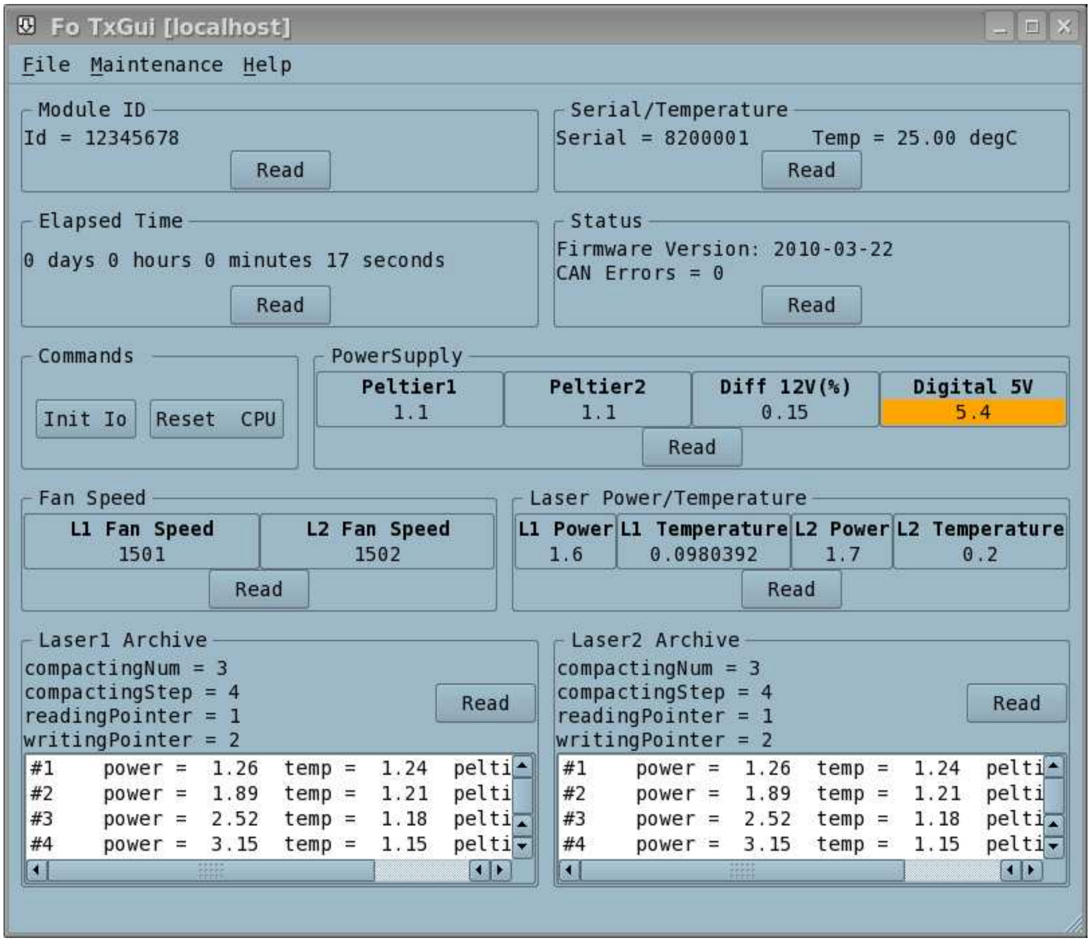
**Syntax:**

```
   fo-tx-gui
```

The software run on antX.iram.fr (X=[1..6]

Procedure to log in `antX.iram.fr`, and execute the program

```
$ ssh –Y backend@ant62.iram.fr
backend@ant62.iram.fr password:
$ fo-tx-gui
```



Module ID
Id = 12345678

Serial/Temperature
Serial = 8200001 Temp = 25.00 degC

Elapsed Time
0 days 0 hours 0 minutes 17 seconds

Status
Firmware Version: 2010-03-22
CAN Errors = 0

Commands

PowerSupply

| Peltier1 | Peltier2 | Diff 12V(%) | Digital 5V |
|---|---|---|---|
| 1.1 | 1.1 | 0.15 | 5.4 |

Fan Speed

| L1 Fan Speed | L2 Fan Speed |
|---|---|
| 1501 | 1502 |

Laser Power/Temperature

| L1 Power | L1 Temperature | L2 Power | L2 Temperature |
|---|---|---|---|
| 1.6 | 0.0980392 | 1.7 | 0.2 |

Laser1 Archive

```
compactingNum = 3
compactingStep = 4
readingPointer = 1
writingPointer = 2
#1    power =  1.26  temp =  1.24  pelti
#2    power =  1.89  temp =  1.21  pelti
#3    power =  2.52  temp =  1.18  pelti
#4    power =  3.15  temp =  1.15  pelti
```

Laser2 Archive

```
compactingNum = 3
compactingStep = 4
readingPointer = 1
writingPointer = 2
#1    power =  1.26  temp =  1.24  pelti
#2    power =  1.89  temp =  1.21  pelti
#3    power =  2.52  temp =  1.18  pelti
#4    power =  3.15  temp =  1.15  pelti
```

With this software, you can

- read all the FO_TX internal values. If the value is out of range, the widget background becomes orange.
- reset the FO_TX microcontroller
- download the Laser EEPROM data, which hold the history.

Note 1:
By default, the software runs in manual mode, i.e. you must click on the *Read* buttons to get the FO_TX values. Nevertheless, the automatic refreshing can be enabled by opening the menu “*Maintenance | Refresh”*

Note 2:
You have to click two times on *Read*, to display the Laser Records
The reason is simple: we cannot know before how many data will be sent by the device. It depends on the EEPROM content, so it is difficult to know when the widget should be updated.

### 5.4 fo-tx-report

Fo-rx-report is a command line program to generate a report about FO RX.

Syntax

```
$ fo-tx-report -h
Fo Tx Report - Print FoTx report
Usage: fo-tx-report [options]
Options:
  -v              Display version information
  -h, -?          Display help
```

Tip: Use redirection to save the program output in a filename.

```
$ fo-tx-report > myreport.txt
```

Example of report

```
############################################################
# Fiber Optic TX - Automatic Report
# Date: 2010-03-22T16:26:18
# Generated on: gre106
############################################################

############################################################
Status
moduleId = 12345678
serial = 136314881
temperature = 25 deg Celcius
Firmware Version = 2010-03-22
Can Error = 0
############################################################

############################################################
Elapsed Time = 0 days 0 hours 4 minutes 43 seconds

############################################################
PowerSupply
Peltier1 = 1.1
Peltier2 = 1.1
Diff 12V(%) = 0.15
Digital 5V = 5.4

############################################################
Laser Power/Temperature
L1 Power = 1.6
L1 Temperature = 0.0980392
L2 Power2 = 1.7
L2 Temperature2 = 0.2

############################################################
Fan Speed
L1 Fan Speed = 1501
L2 Fan Speed = 1502

############################################################
Laser1 Archive
compactingNum = 3
compactingStep = 4
readingPointer = 1
writingPointer = 2
#1    1.26  1.24  0.04
#2    1.89  1.21  0.07
[...]

# End of Report
```

# 6 Troubleshooting

## 6.1 CAN maintenance

See document IRAM-COMP-057 *CanIp*

## 6.2 Modules

You can check if the device drivers are loaded:

```
$ /sbin/lsmod | grep tpmc
Module                  Size  Used by
tpmc816drv             10704  4
```

If this module is missing, it means that there was a problem during the computer initialization. You should restart the computer with the ‘reboot’ command.

## 6.3 Processes

You can list the current processes and check that all the required processes are present:

```
$ ps –f –u oper
```

Look for the following processes:

```
oper      1572  1571  0 Mar26 ?        00:01:01
/home/introot/fo/bin/release/CanManager -d=/dev/tpmc816_1 -p=2501 -l=50
oper      1599     1  0 Mar26 ?        00:01:25 fo-rx-server
```

If one of these processes is missing, it means that it has crashed. If it happens, please contact the IRAM computer group. And then you can reboot the computer.