IRAM-COMP-078

Revision: 0
2011-08-01

Contact Author

Institut de RadioAstronomie Millimétrique

Subreflector software

Owner Sebastien Blanchet

Keywords: subreflector, software

Approved by: A.Perrigouard	Date: January 2011	Signature:

Change Record

REVISION	DATE	AUTHOR	AFFECTED SECTION/PAGE	REMARKS

Content

1 Introduction

This document is the user manual for the subreflector software on the IRAM Plateau de Bure Interferometer.

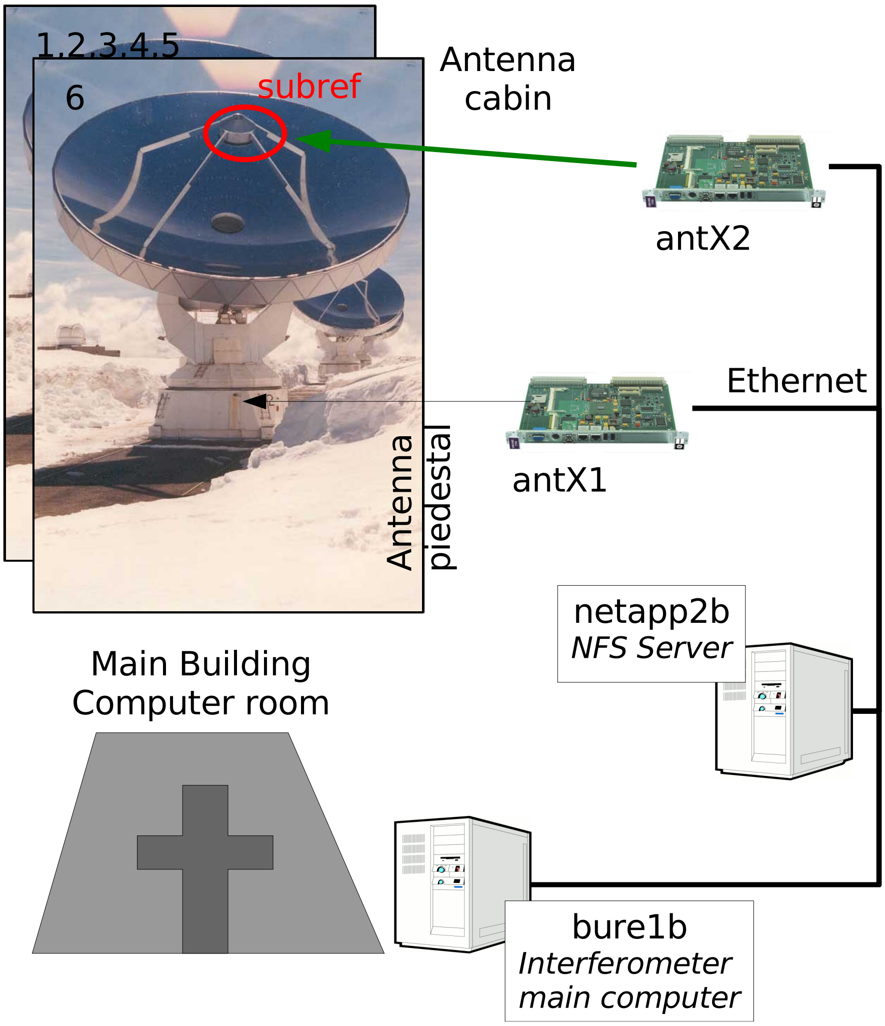
When the antenna elevation changes, the antenna paraboloid deforms with its own weight, but thanks to its special design, it keeps a parabolic form. Nevertheless, the focal point moves with the elevation, so the subreflector mirror must also be moved with the elevation.

2 Architecture

The subreflector motors are driven through a VME board, plugged in the VME crate of the cabin.

See IRAM-COMP-021 for a complete description of the VME subreflector hardware.

T pedestal computer (antX1) computes the antenna elevation and new subreflector position. Then it sends a network request to the cabin computer (antX2) to move the subreflector by driving the VME subreflector board.



3 Requirements

3.1 Hardware requirement

The subreflector software runs only on a VME single board computer with a Tundra Universe 2 chip, because it needs a VME interface to drive the subreflector VME board.
At Plateau de Bure, we use a VMIVME- 7700 from GeFanuc Automation

Figure 1: VMIVME-7700 from GEFanuc Automation

But for the test environment, any PC can be used, because a CAN bus simulator is provided

3.2 Software requirements

The software targets Linux Fedora Core 4 i386 as main platform, but it should run on any computer with Linux 2.6.x

The following software are required to build and run the programs:

Name	Description	Version	Download
g++	GNU C++ compiler	>= 4.0.2	http://gcc.gnu.org
Qt	C++ framework	>= 4.7.1	http:/qt.nokia.com
Sqlite	Embedded SQL database	>= 3.3.6	http://sqlite.org
subversion	Version control system	>= 1.5	http://subversion.tigris.org
doxygen	Automatic documentation system	>= 1.6.1	http://doxygen.org

All these software are very common, and have ready-to-install packages for your favorite Linux distribution.

4 Installation

4.1 Build receiver software

Extract code from the repository

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

Then use the Makefile, to extract automatically the dependencies

```
$ cd subref
$ qmake
$ make get_deps
```

Build the subreflector code

```
$ ./build.sh
```

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

```
$ make doc
```

To install the software **and the default data** in `/home/introot/subref`
Note: the shared libraries are installed in `/home/introot/lib`

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

Warning:
If you wish to update only the software **without reinstalling the default data**, use `./install.sh`(without the *all*). It is equivalent to run `./install.sh programs`

If you are not sure, it is safer to backup `/home/introot/subref` first

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

4.2 Configure environment

You should add `/home/introot/subref/bin` to your PATH

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

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

5 Internal programs

This section describes internal programs that drive the receiver. These programs are executed automatically, therefore normal users are note expected to run them.

5.1 CanManager

For a complete description of the the CanManger, see the document *can-ip.pdf*
For the subref software, CanManager must listen on ports 2502

```
$ CanManager –p=2502
```

5.2 subref-control_legacy

subref-control_legacy drives the subreflector board.
It is designated as legacy, because we expect to rewrite latter this application to support the XML-RPC protocol instead of its own old binary protocol.

5.2.1 Internal description

This application has 3 threads:

1. the network thread
2. the VME thread
3. the CAN thread

5.2.1.1 The network thread

This thread runs the procedure `listenNetwork()`
This procedure opens a TCP socket on port 1052 and wait for network request.

Protocol description
It receives a SubRefRequest_t data structure

```
#define MOTOR_NUM 5
typedef struct {
   short request;    /* boolean to init motor         */
   short rpos[MOTOR_NUM]; /* request position for each motor*/
} SubRefRequest_t;
```

If request is not null, the first 4 motors will be initialized (by an other thread)
Note: the last motor (#5) is not initialized here. This motor is set up manually in a different way

The thread sends back the board status:

```
typedef struct {
   short status_reg; /* status register */
   short apos[MOTOR_NUM];   /* actual position for each motor */
   short reqt[MOTOR_NUM];   /* request position for each motor */
   short cmnd[MOTOR_NUM];    /* command register (3-bit) */
   short stus[MOTOR_NUM];    /* status register  (3-bit) */
} SubRefStatus_t;
```

5.2.1.2 The VME thread

This thread runs the procedure `subrefLoop()`

There is basically a finite-state machine for each motor.
For legacy reason, this FSM is very complex, but it can be modified without breaking compatibility with client programs. Therefore we must keep this FSM as is, until the program and all its clients are rewritten.

5.2.1.3 the CAN thread

This thread listens for CAN messages. It runs the procedure `listenCan()`

Why there is a CAN interface?
When the new receivers has been designed, we planned to use only the CAN bus for the long-term. But we do not have time and money to redesign the subreflector VME board.

Nevertheless it was a good idea to have only one bus for all the receiver: the software development becomes simpler and more regular. Therefore, I have added a virtual CAN interface to the subreflector program, so the client programs can see it as a CAN device through a virtual CAN bus.

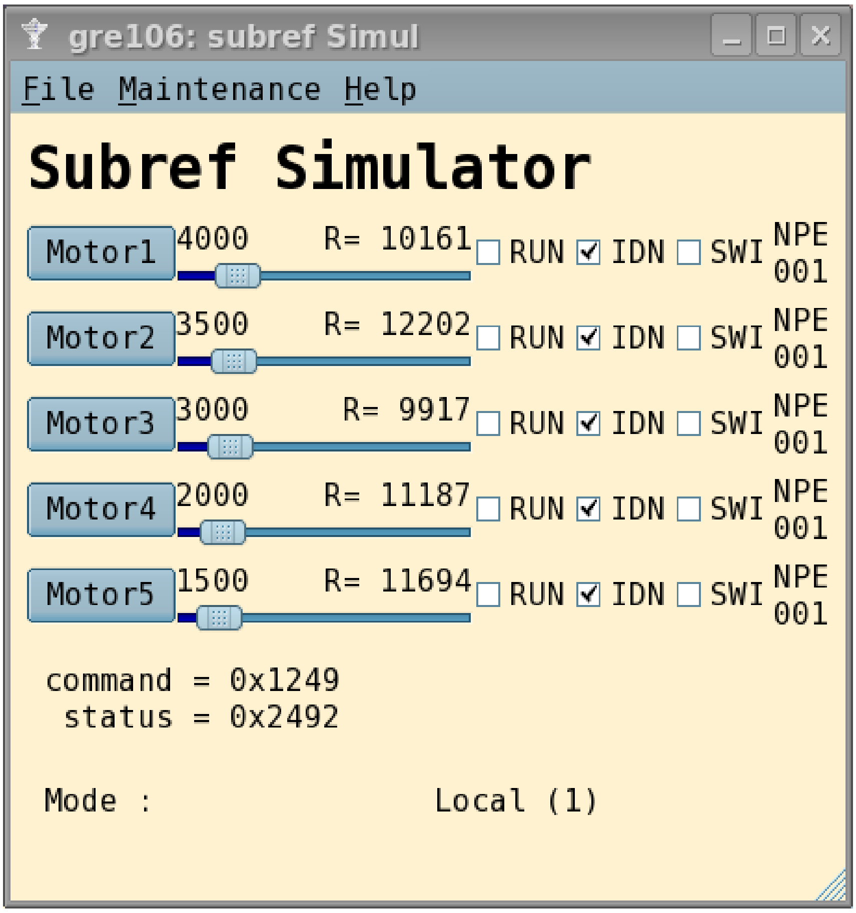
5.2.2 Syntax

```
$ subref-control_legacy
```

There are neither options nor online help.

5.3 Simulator

The goal of this program is to simulate the subreflector, from the CAN point of view, so that the other software can be written before the receiver hardware is ready.



Subref Simulator

Motor1 4000 R= 10161 □ RUN ☑ IDN □ SWI NPE 001

Motor2 3500 R= 12202 □ RUN ☑ IDN □ SWI NPE 001

Motor3 3000 R= 9917 □ RUN ☑ IDN □ SWI NPE 001

Motor4 2000 R= 11187 □ RUN ☑ IDN □ SWI NPE 001

Motor5 1500 R= 11694 □ RUN ☑ IDN □ SWI NPE 001

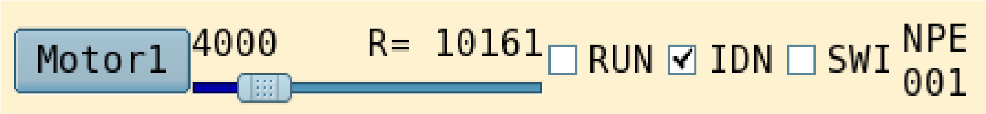
```
command = 0x1249
 status = 0x2492
```

Mode : Local (1)

5.3.1 Syntax

```
$ subref-simul
```

5.3.2 Usage



Motor1 4000 R= 10161 □ RUN ☑ IDN □ SWI NPE 001

Motor1: it is the motor name
4000: the actual position

> The simulator modifies the actual position to simulate the motor movement. Nevertheless you can change the actual position with the slider or by clicking on the motor name and then entering a numerical value.

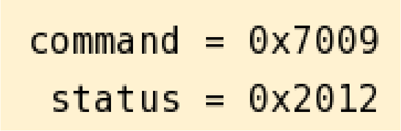
R=10161: the requested position

RUN, IDN, SWI are the 3 status bits on the motor

Command Bit	Description
RUN	Running
IDN	Init done
SWI	Motor Switch

Status Bit	Description
N	Negative Velocity Request: used to move the motor back.
P	Positive Velocity Request: used to move the motor forward.

E	Enabled: to enable/disable the position mode



```
command = 0x7009
 status = 0x2012
```

Aggregation of all motor command bits (and status bits) into two words.

6 User programs

This section describes the user programs. These programs are installed in `/home/introot/subref/bin`.
For convenience, all program names starts with the prefix `subref-`
For the software developers, there are debug versions of these programs: just add the suffix `.Debug` to the program name.
The data files are installed in `/home/introot/subref/data`

6.1 subref-dump

This program dumps the shared memory of the subreflector. So we can inspect that happens in subreflector-control, without disturbing it.

6.1.1 Syntax

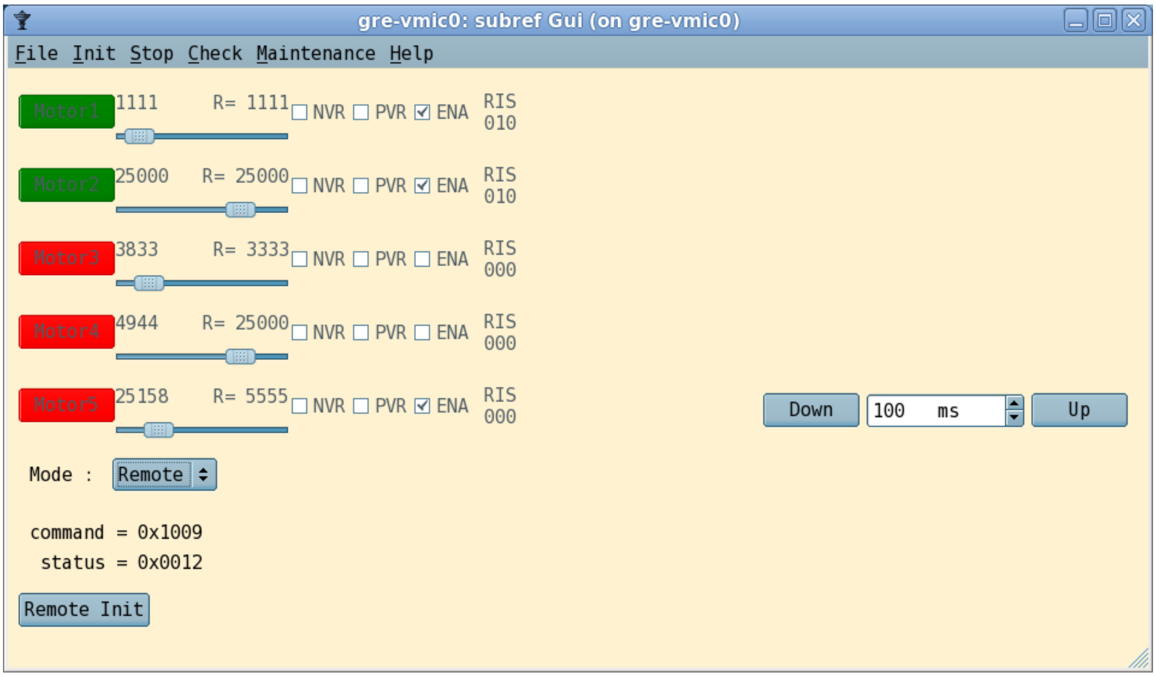
```
$ subref-dump -h
Subref shared memory dumper
Dump the subref shared memory
Usage: subref-dump [options]
Options:
  -v        Display version information
  -h, -?    Display help
  -l        Repeat dump every second, for ever
```

6.1.2 Examples

```
$ subref-dump
Dump Subref shared memory (0)
======================================
msgCounter = 50799
request = 0  ; rpos 7285 7362 7285 2865 0
commandRegister = 0x6249
manualMode = 0x0
status_reg = 0x0
      apos    reqt cmnd stus
#1:   7285    4    1    0
#2:   7362    4    1    0
#3:   7285    4    1    0
#4:   2865    4    1    0
#5:      0    1    6    0
```

6.2 subref-gui

This program is designed to drive manually the subreflector motors.


gre-vmic0: subref Gui (on gre-vmic0)
File Init Stop Check Maintenance Help
Down
Up
Remote Init

Motor1 1111 R= 1111 □ NVR □ PVR ☑ ENA RIS 010

Motor2 25000 R= 25000 □ NVR □ PVR ☑ ENA RIS 010

Motor3 3833 R= 3333 □ NVR □ PVR □ ENA RIS 000

Motor4 4944 R= 25000 □ NVR □ PVR □ ENA RIS 000

Motor5 25158 R= 5555 □ NVR □ PVR ☑ ENA RIS 000

100 ms

Mode : Remote

command = 0x1009

status = 0x0012

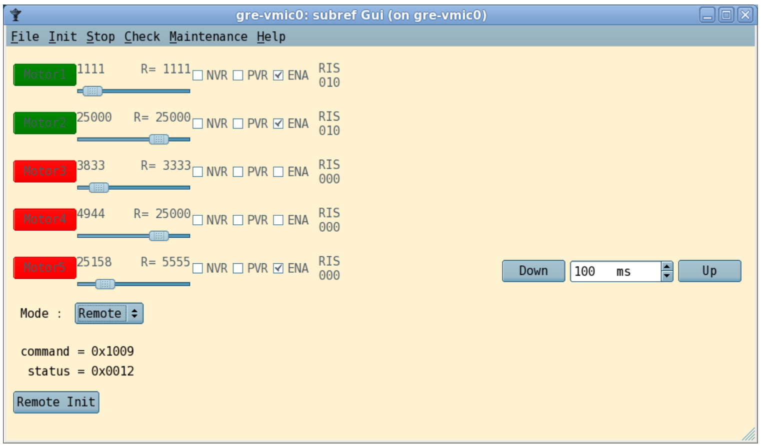
6.2.1 Syntax

```
$ subref-gui
```

6.2.2 Usage

Modes

There are 2 modes: Local and Remote


gre-vmic0: subref Gui (on gre-vmic0)
File Init Stop Check Maintenance Help
Down
Up
Remote Init

Motor1 1111 R= 1111 □ NVR □ PVR ☑ ENA RIS 010

Motor2 25000 R= 25000 □ NVR □ PVR ☑ ENA RIS 010

Motor3 3833 R= 3333 □ NVR □ PVR □ ENA RIS 000

Motor4 4944 R= 25000 □ NVR □ PVR □ ENA RIS 000

Motor5 25158 R= 5555 □ NVR □ PVR ☑ ENA RIS 000

100 ms

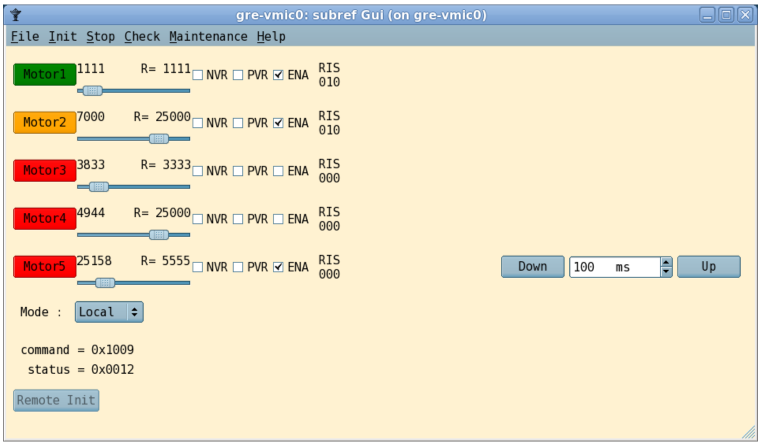
Mode : Remote

command = 0x1009

status = 0x0012

In remote mode, only the program subref-control controls the motors. So their widgets are disabled.

But the *Remote Init* button is enabled.


gre-vmic0: subref Gui (on gre-vmic0)
File Init Stop Check Maintenance Help
Down
Up
Remote Init

Motor1 1111 R= 1111 □ NVR □ PVR ☑ ENA RIS 010

Motor2 7000 R= 25000 □ NVR □ PVR ☑ ENA RIS 010

Motor3 3833 R= 3333 □ NVR □ PVR □ ENA RIS 000

Motor4 4944 R= 25000 □ NVR □ PVR □ ENA RIS 000

Motor5 25158 R= 5555 □ NVR □ PVR ☑ ENA RIS 000

100 ms

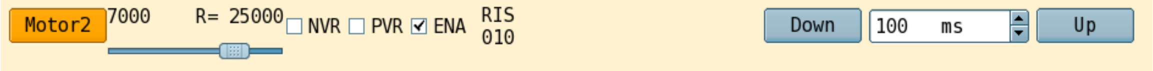
Mode : Local

```
command = 0x1009
 status = 0x0012
```

In local mode, only the program subref-gui controls the motors. So their widgets are enabled.

But the *Remote Init* button is disabled.

Motor description



Motor2 7000 R= 25000 □ NVR □ PVR ☑ ENA RIS 010 100 ms

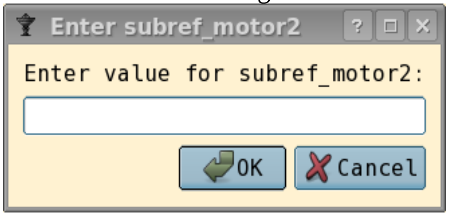
From left to right

Motor2: it is a push button with the motor name. It can take 3 colors:

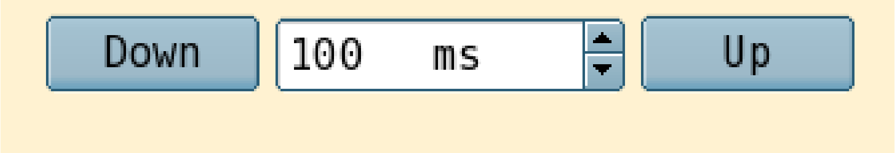
- Green: the motor is initialized and has reached its requested position
- Orange: the motor is initialized but it has not yet reached it requested position
- Red: the motor is not initialized

7000: it is the current position

R =25000: it is the requested position. You can change it by moving the slider behind or by clicking on the motor name and entering a numerical value.



Enter value for subref_motor2:



100 ms

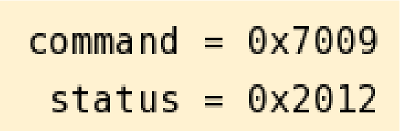
The *Up* and *Down* buttons are used so send a pulse order to the motor. The motor will move in the specified direction (Up/Down) during the specified time (in milliseconds).

This functionality is useful to drive a motor without coder. (For example actually the motor 5 have not coder).

NVR, PVR, ENA are the 3 command bits on the motor

Command Bit	Description
NVR	Negative Velocity Request: used to move the motor back.
PVR	Positive Velocity Request: used to move the motor forward.
ENA	Enabled: to enable/disable the position mode

Status Bit	Description
R	Running
I	Init done
S	Motor Switch



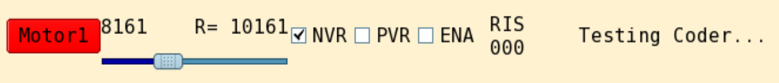
command = 0x7009
status = 0x2012

Aggregation of all motor command bits (and status bits) into two word.

This push button is used to simulate a remote init request from the pedestal computer. So it must be used in

Menus
Menu Init: use this menu to initialize the motors
Menu Stop: use this menu to stop the motors
Menu Check: use this menu to check the motor

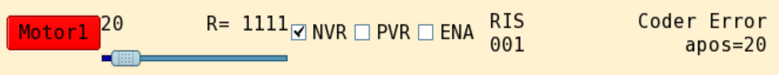
Checking example:



Motor1 8161 R= 10161 ☑ NVR □ PVR □ ENA RIS 000 Testing Coder...

Checking the coder

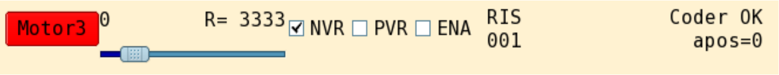
Result



Motor1 20 R= 1111 ☑ NVR □ PVR □ ENA RIS 001 Coder Error apos=20

The Coder has a problem (apos=0)

or



Motor3 0 R= 3333 ☑ NVR □ PVR □ ENA RIS 001 Coder OK apos=0

The Coder is OK (apos=0)

6.3 subref-check

This program is used to check the subreflector coder.

6.3.1 Syntax

```
$ subref-check -h
subref-check
Check subreflector coders
Usage: subref-check [options]
Options:
  -v          Display version information
  -h, -?      Display help

  -c=N        Number of cycles. Default 100
  -p=p1,p2    Extrema position for the cycle
  -m=motor_list     Comma-separated list of motor number to check

Example: run 100 cycles between positions [333, 999], for motors 1 and 2
  subref-check -c=100 -p=333,999 -m=1,2
```

6.3.2 Example

```
$ subref-check -c=5 -p=200,10000 -m=1,2
Cycles = 10
Positions=200,10000
motors = 1,2
Initialize motor #1
```

```
Searching Zero for motor 1
Initialize motor #2
Searching Zero for motor 2
Wait for motors initialization... OK
Iteration #1
Iteration #2
Iteration #3
Iteration #4
Iteration #5
Checking coder for motor 1
Checking coder for motor 2
******************************************
Coder status
******************************************
Motor 1: apos=20 => Coder OK
Motor 2: apos=10 => Coder OK
Do you want to restore the normal state ? y/[n] : y
Initialize motor #1
Searching Zero for motor 1
Initialize motor #2
Searching Zero for motor 2
Wait for motors initialization... OK
```