INTERNAL REPORT

SRT Antenna Control: a graphic tool for setting and pointing the Sardinia Radio Telescope

Alessandro Corongiu

Report N. 40, released: 22/09/2014

Reviewer: N. D'Amico, M. Murgia

OAC Osservatorio Astronomico di Cagliari

Contents

Chapter 1

Introduction

The Sardinia Radio Telescope[1] (SRT) is a brand new observing facility located about 35 km north of Cagliari, the main city of Sardinia. It has been designed for covering a frequency range from 300 MHz up to 100 GHz, by hosting up to 13 receivers in various focal positions, and for covering a wide set of topics in radio astronomy. The telescope controlling software, called `NURAGHE`[2], has been developed under the Alma Common Software[3] (ACS) environment, and allows the whole control of the observing sessions, from the antenna setup to the source tracking and the data acquisition.

`NURAGHE` operations are quite involved since its user interface is a command line terminal and all necessary procedures require a sequence of manual commands. This statement is particularly true in the case of the antenna setup and source tracking. On the other side `NURAGHE` offers the possibility of being operated by an external client connected via a dedicated socket port. Thanks to this feature, a tool has been designed for simplifing, from the user's point of view, the procedures for the antenna setup and source tracking. This tool, named SRT Antenna Control (SAC), has been developed under the `Seadas Developement System` (see OAC internal report[4] number 37, hereafter report37), is based on a graphic user interface and can be used both as a stand alone tool or inside a wider application for managing observing sessions with the Sardinia Radio Telescope.

This report illustrates all aspects of SRT Antenna Control. It is addressed to both the user that wishes to use this tool for controlling the Sardinia Radio Telescope, and the developer that wishes to implement it inside a more complex application. The report is organized as follows. Chapter 2 illustrates the graphic user interface and all secondary windows, while chapter 3

[1] `http://www.srt.inaf.it`
[2] Orlati et al. 2012, SPIE, 8451, 2
[3] `http://www.eso.org/projects/alma/develop/acs/`
[4] `http://www.oa-cagliari.inaf.it/area.php?page_id=10&skip=3`

ilustrates the setup file and the syntax for all keywords. Chapter 4 illustrates the programming of this tool and is addressed to developers only, who are asked to have previously read report37 and the latest version of the `NURAGHE` manual.

Chapter 2

The graphic interface

`SRT Antenna Control` is a graphic tool for easily setting and displaying all necessary parameters for the antenna setup and source tracking. Its main and auxiliary widgets, displayed in figures 2.1 & 2.2, maintain the default widget organization for an `SDSAntenna` object (see report37). In the main widget it has been added a combo box for selecting the shape and behaviour of the main mirror's active surface, and frames have been implemented for setting the necessary parameters for the `L-P` dual band and `C` band receivers.

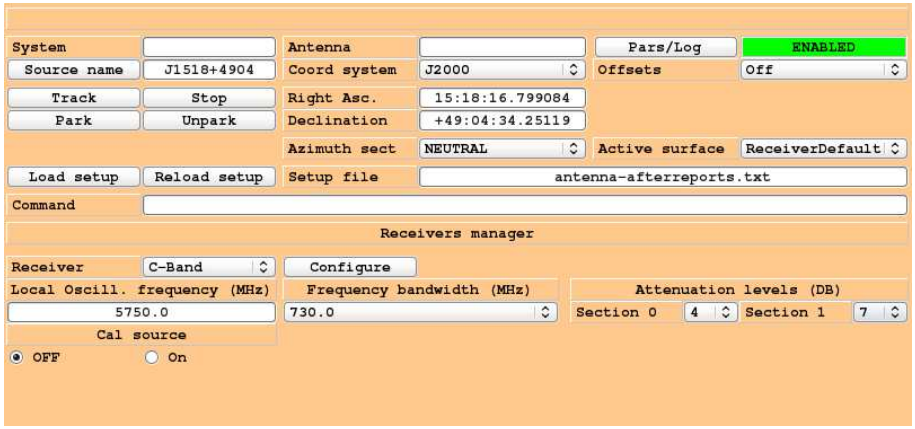


Figure 2.1: Main widget layout

The combo box for setting the active surface is located at the widget's right side, is identified by the label `Active surface`, and allows the following options:

0) `ReceiverDefault`: this option selects the default shape for the selected receiver, namely *ParabolicFixed* for primary focus receivers, *Shaped* for gregorian focus and beam wave guide focus receivers;

1) `Parabolic`: this option selects the *Parabolic* shape and enables the

elevation corrections.

2) `ParabolicFixed`: this option selects the *Parabolic* shape, optimized for an elevation of 45°.

3) `Shaped`: this option selects the *Shaped* shape and enables the elevation corrections.

4) `ShapedFixed`: this option selects the *Shaped* shape, optimized for an elevation of 45°.

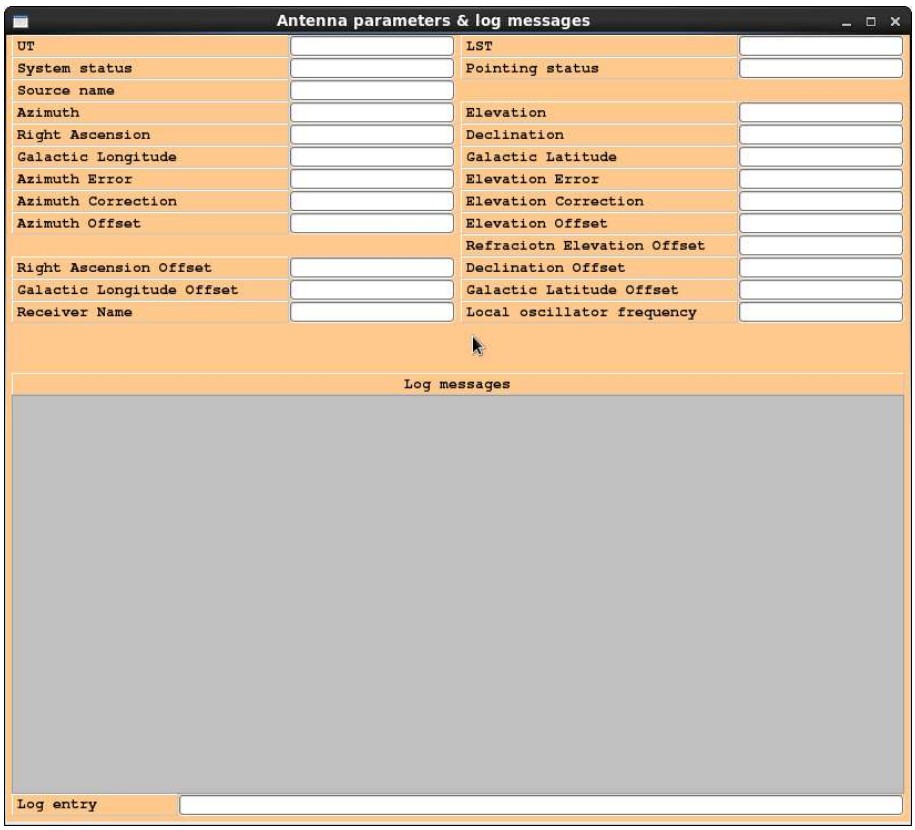


Figure 2.2: Auxiliary window layout

The auxiliary window layout, displayed in figure 2.2 is the `SDSAntenna` default one (see report37), since it hasn't been necessary the adding of further elements.

In the receiver's management frame, the combo box that allows the receiver's selection, identified by the label `Receiver`, contains the following options:

1: `P-Band`, for selecting the P-Band receiver;

2: `L-Band`, for selecting the L-Band receiver;

3: `LP-Dual`, for selecting the P-Band and L-Band receivers to be simultaneously used (functionalities not yet implemented);

4: `S-Band`, for selecting the incoming S-Band receiver (functionalities not yet implemented);

5: `C-Band`, for selecting the C-Band receiver;

6: `K-Band`, for selecting the K-Band multibeam receiver (functionalities not yet implemented);

When an item is selected, the frame is shown for displaying the fields that allow the parameters' setting. The frame for the `C-Band` receiver, displayed in figure 2.3, contains objects for setting the local oscillator frequency and the frequency bandwidth, both espressed in MHz, the attenuation levels for the two signal sections and two radio buttons for switching off and on the calibration source.

Figure 2.3: C-Band widget layout

Frames for the `P-Band` and `L-Band` receivers are displayed simultanously if the item `LP-Dual` is selected in the `Receiver` combo box (figure 2.4), otherwise only the `P-Band` frame (figure 2.6) or the `L-Band` (figure 2.5) are shown accordingly to the selection. In both frames combo boxes are present for selecting the polarization to be detected, the frequency bandwidth to be observed and the attenuation levels for the two signal sections.

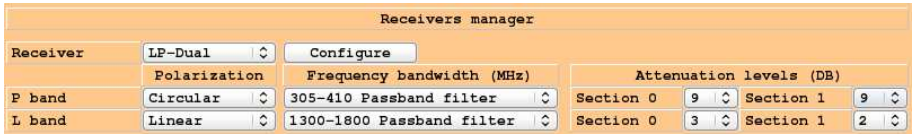


Figure 2.4: LP-Dual widget layout

Figure 2.5: L-Band widget layout

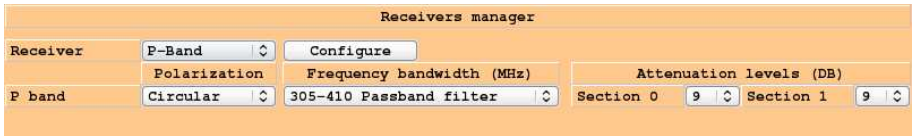
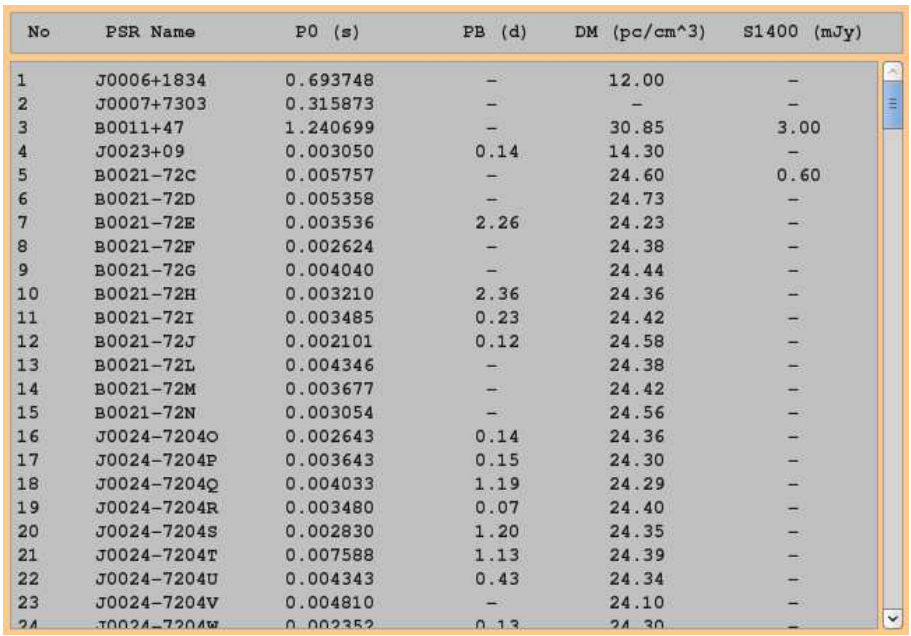


Figure 2.6: P-Band widget layout

The source catalogue (figure 2.7) contains all radio pulsar present in the `ATNF` catalogue `psrcat`[1]. Each pulsar is identified by its catalogue name and is displayed with its spin and orbital period, its dispersion measure and its flux at 1400 MHz. A source is selected by clicking with the mouse left button on any point of its displayed line. Its name and coordinates are automatically set in the dedicated objects in the main widget.



No	PSR Name	P0 (s)	PB (d)	DM (pc/cm^3)	S1400 (mJy)
1	J0006+1834	0.693748	–	12.00	–
2	J0007+7303	0.315873	–	–	–
3	B0011+47	1.240699	–	30.85	3.00
4	J0023+09	0.003050	0.14	14.30	–
5	B0021-72C	0.005757	–	24.60	0.60
6	B0021-72D	0.005358	–	24.73	–
7	B0021-72E	0.003536	2.26	24.23	–
8	B0021-72F	0.002624	–	24.38	–
9	B0021-72G	0.004040	–	24.44	–
10	B0021-72H	0.003210	2.36	24.36	–
11	B0021-72I	0.003485	0.23	24.42	–
12	B0021-72J	0.002101	0.12	24.58	–
13	B0021-72L	0.004346	–	24.38	–
14	B0021-72M	0.003677	–	24.42	–
15	B0021-72N	0.003054	–	24.56	–
16	J0024-7204O	0.002643	0.14	24.36	–
17	J0024-7204P	0.003643	0.15	24.30	–
18	J0024-7204Q	0.004033	1.19	24.29	–
19	J0024-7204R	0.003480	0.07	24.40	–
20	J0024-7204S	0.002830	1.20	24.35	–
21	J0024-7204T	0.007588	1.13	24.39	–
22	J0024-7204U	0.004343	0.43	24.34	–
23	J0024-7204V	0.004810	–	24.10	–
24	J0024-7204W	0.002352	0.13	24.30	–

Figure 2.7: Catalogue window

[1] `http://www.atnf.csiro.au/people/pulsar/psrcat/`

Chapter 3

The setup file

The `SRT Antenna Control` setup file is structured accordingly to the general prescription for a generic `SDSDevice` setup file, namely entries are organized in rows whose generic syntax is:

```
[keyword] = [value(s)]
```

Here below is shown an example of a complete setup file. Entries can be sorted in any order, with the only exception of the `CalSource` keyword, that must be always placed after the `Receiver` keyword:

```
Source = B0833-45,J2000,08:35:10.742,-45:10:27.987
AzSector = CW
ActiveSurface = ShapedFixed
Offsets = Az/El,2.0,-3.0
Receiver = C-Band,5750.0,730.0,4,7
CalSource = On
```

3.1 Keywords syntax

ActiveSurface
This keyword allows to set the shape and behaviour for the primary mirror's active surface. Its syntax is:

```
ActiveSurface = [surface mode]
```

where `[surface mode]` can be *Parabolic* for the adaptive parablic mode, *ParabolicFixed* for the fixed parabolic mode, *Shaped* for the adaptive shaped mode, *ShapedFixed* for the fixed shaped

mode, and *ReceiverDefault* for the default mode assigned to the selected receiver, namely *ParabolicFixed* for primary focus receivers, *Shaped* for gregorian focus and beam wave guide focus receivers.
Example: `ActiveSurface = ShapedFixed`

AzSector
This keyword allows to set the antenna azimuth sector for tracking the source. Its syntax is:

`AzSector = [sector ID]`

where `[sector ID]` can be `CW` for the *clockwise* sector, `CCW` for the *counterclockwise* sector, or `NEUTRAL` for allowing the antenna system to determine the most useful sector.
Example: `AzSector = CCW`

CalSource
This keyword allows to switch on and off the receiver's calibration source. Its syntax is:

`CalSource = [ON/OFF]`

Example: `CalSource = ON`

The line containing this keyword **must** be always placed **after** the line for setting up the receiver.

Offsets
This keyword allows to set the poiting offsets with respect to the source coordinates. Two syntaxes are allowed:

1) `Offsets = OFF`
   Pointing offsets are disabled and the source is pointed at the indicated coordinates.

2) `Offsets = [frame],[longitude offset],[latitude offsets]`
   Pointing offsets are set in the frame `[frame]` (values: `Az/El` for the horizontal frame, `J2000` for the celestial frame, `Galactic` for the galactic frame) and are equal to `[longitude offset]` and `[latitude offsets]` respectively along the longitude and latidude coordinate in the frame

selected for the offsets. Offsets values are always in decimal degrees.
Example: `Offsets = J2000,+3.0,-5.0`

Receiver

This keyword allows to select the receiver and set its configuration parameters. Its general syntax is:

`Receiver = [receiver name],[receiver parameters]`

where `[receiver name]` is the name of the selected receiver, and `[receiver parameters]` is a list of comma separated strings that indicate the value for each parameter. The number and meaning of each parameter depend on the selected receiver.

`P-Band` *syntax*:

`Receiver = P-Band,[band number],[polarization],[sect 0 att],[sect 1 att]`

`[band number]` is an integer that identifies the selected frequency band:

1 = 305 MHz−410 MHz (no filter)

2 = 310 MHz−350 MHz

3 = 305 MHz−410 MHz (pass band filter)

`[polarization]` is the polarization to detect, namely `Circular` or `Linear`.
`[sect 0 att]` and `[sect 1 att]` are the attenuations in decibel for the two polarizations separately, expressed as integer numbers from `0` to `15`.
Example: `Receiver = P-Band,2,Linear,3,7`

`L-Band` *syntax*:

`Receiver = L-Band,[band number],[polarization],[sect 0 att],[sect 1 att]`

`[band number]` is an integer that identifies the selected frequency band:

1 = 1300 MHz−1800 MHz (no filter)

2 = 1320 MHz−1780 MHz

3 = 1350 MHz−1450 MHz (VLBI)

4 = 1300 MHz−1800 MHz (pass band filter)

5 = 1625 MHz−1715 MHz (VLBI)

`[polarization]` is the polarization to detect, namely `Circular` or `Linear`.
`[sect 0 att]` and `[sect 1 att]` are the attenuations in decibel for the two polarizations separately, expressed as integer numbers from `0` to `15`.
Example: `Receiver = L-Band,4,Circular,2,1`

`LP-Dual` *syntax*:

```
Receiver = LP-Dual,[P band number],[P polarization],[P sect 0 att],[P sect 1 att],
[L band number],[L polarization],[L sect 0 att],[L sect 1 att]
```

parameters `[P band number]`, `[P polarization]`, `[P sect 0 att]` and `[P sect 1 att]` are respectively `[band number]`, `[polarization]`, `[sect 0 att]` and `[sect 1 att]` for the `P` band and assume the values indicated for the `P-Band` receiver, while parameters `[L band number]`, `[L polarization]`, `[L sect 0 att]` and `[L sect 1 att]` are again `[band number]`, `[polarization]`, `[sect 0 att]` and `[sect 1 att]` for the `L` band and assume the values indicated for the `P-Band` receiver.
Example: `Receiver = LP-Dual,2,Circular,8,6,4,Linear,11,11`

`C-Band` *syntax*:

```
Receiver = C-Band,[local oscillator frequency],[bandwidth],[sect 0 att],[sect 1 att]
```

`[local oscillator frequency]` is the frequency, in MHz, of the receiver's local oscillator.
`[bandwidth]` is the frequency band width, in MHz, whose allowed values are 300.0, 730.0, 1250.0, 2000.0
`[sect 0 att]` and `[sect 1 att]` are the attenuations in decibel for the two polarizations separately, expressed as integer numbers from `0` to `15`.
Example: `Receiver = C-Band,6300.0,1250.0,9,9`

Source

This keyword allows to set the source name and its coordinates. The following syntaxes are allowed:

1) `Source = [name]`
   The source whose name is `[name]` is set with its catalogue coordinates in the currently set coordinate system.
   Example: `Source = B1937+21`

2) `Source = [name],[coordinate system]`
   The source whose name is `[name]` is set with its catalogue coordinates in the coordinate system indicated by `[coordinate system]` (values: `Az/El` for the horizontal frame, `J2000` for the celestial frame, `Galactic` for the galactic frame).
   Example: `Source = J1910-5959C,Galactic`

3) `Source = [name],[coordinate system],[longitude],[latitude]`
   The source whose name is `[name]` is set with coordinates `[longitude]` `latitude` in the coordinate system indicated by `[coordinate system]`.
   Example: `Source = B0833-45,J2000,08:35:10.742,-45:10:27.987`

Syntaxes `1` and `2` can be used for catalogue sources only, while syntax `3` is mandatory for objects not present in the catalogue.

Chapter 4

Technical description

SRT Antenna Control has been designed accordingly to the prescription illustrated in report37 for implementing an `SDSAntenna` object for controlling a specific antenna, namely:

1 Class `SDSCatalogue` has been subclassed to the new class `PulsarCatalogue` for implementing all `SDSCatalogue` virtual slots accordingly to the catalogue format.

2 Class `SDSCommThread` has been subclassed to the new class `SRTCommThread` for implementing all `SDSCommThread` virtual slots accordingly to the peculiar SRT procedures for setting the antenna and pointing a source.

3 Class `SDSAntenna` has been subclassed to the new class `SRT` for implementing all `SDSAntenna` virtual slots accordingly to the peculiar SRT procedures for setting the antenna and pointing a source. `SDSAntenna` members `SDSCommThread commThread` and `SDSCatalogue sourceCat` have been recasted to the new classes `SRTCommThread` and `PulsarCatalogue` respectively, and all necesary signal slot connections have been established between the newly subclassed members and the `SRT` object.

4.1 Class PulsarCatalogue

Class `PulsarCatalogue` inherits from class `SDSCatalogue`, and its implementation consists in the reimplementation of the two `SDSCatalogue` virtual functions, namely `QString PulsarCatalogue::findSourceCoords(QString name)`

for obtaining the pulsar coordinates starting from the pulsar name as input, and `QString PulsarCatalogue::getSourceParameters(QString catLine)` for obtaining the same parameters by starting from the clicked line in the catalogue window.

Inheritance

Inherits from: `SDSCatalogue`

Public slots

`QString PulsarCatalogue::findSourceCoords(QString name)`
This slot extracts from the pulsar catalogue `psrcat` the pulsar coordinates in the already set coordinate system, and returns them in the form [`pulsarname longitude latitude`].

`QString PulsarCatalogue::getSourceParameters(QString catLine)`
This slot extracts the pulsar name indicated in its argument string, then calls `QString PulsarCatalogue::findSourceCoords(QString name)` for obtaining the source coordinates, and returns the obtained string.

4.2 Class SRTCommThread

Class `SRTCommThread` inherits from class `SDSCommThread`, and its implementation consists in the reimplementation of the three `SDSCommThread` virtual functions, namely `QString SRTCommThread::composeCommand(QString cmd)` for reformatting the command to be sent and returning it in the requested format, `void SRTCommThread::processMessage(QString msg)` for taking the basic decisions whenever a messages is received, and `void SRTCommThread::setAwaitingCommand()` for storing those commands whose execution needs to be confirmed before sending to the antenna system the following command in a procedure. The second and third mentioned slots have a preliminary implementation only, since the regular use of `SRT Antenna Control` may highlight those situations that require specific actions.

Inheritance

Inherits from: `SDSCommThread`

Public slots

`QString SRTCommThread::composeCommand(QString cmd)`
This slot returns its argument string, since the `NURAGHE` system does not require any command formatting by clients connected through the dedicated socket port.

`void SRTCommThread::processMessage(QString msg)`
This slot is temporarily implemented for reacting whenever a command has been refused from `NURAGHE`. In these cases the procedure is interrupted, by setting to `none` the string stored in member `QString commandQueue`, and the signal `void SDSCommThread::newMessage(QString)` is emitted a first time for passing to the `SRT Antenna Control` main object the received message, and a second time for passing the string `Procedure aborted` which will be displayed in the log window.

`void SRTCommThread::setAwaitingCommand()`
This slot tempoarily contains an empy implementation. The regular use of `SRT Antenna Control` will suggest those command to be set as awaiting.

4.3 Class SRT

Class `SRT` provides the `SRT Antenna Control` main object. This class has been implemented by subclassing the `SDSAntenna` class for reimplementing all `SDSAntenna` virtual slots, for recasting members `SDSCommThread *commThread` and `SDSCatalogue *sourceCat` to the related fully implemented subclasses, namely `SRTCommThread` and `PulsarCatalogue` respectively, for establishing all necessary signal/slot connections, and finally for adding all necessary objects for managing all SRT peculiar parameters and receivers.

Parameter objects introduced in this class are:

a) `SDSCombo ActiveSurface` for selecting the primary mirror shape and behaviour;

b) SDSEdit CLOfreq, SDSCombo Cbw, SDSCombo CSec0 and SDSCombo CSec1 for managing, for the C-Band receiver, the local oscillator frequency, the frequency band width and the attenuation levels for the two signal sectors;

c) SDSCombo LpolsCb, SDSCombo LbandsCb, SDSCombo LSec0 and SDSCombo LSec1 for managing, for the L-Band receiver, the polarization to be detected, the band to be observed and the attenuation levels for the two signal sectors;

d) SDSCombo PpolsCb, SDSCombo PbandsCb, SDSCombo PSec0 and SDSCombo PSec1 for managing, for the P-Band receiver, the polarization to be detected, the band to be observed and the attenuation levels for the two signal sectors;

Two radio buttons, QRadioButton CcalOn and QRadioButton CcalOff, organized in the button group QButtonGroup CcalBg, allow to switch on and off the calibration source for the C-Band receiver, and some QLabel objects have been implemented for hosting all necessary descriptions.

Objects for managing the receivers' parameters have been set as child of QWidget objects: QWidget Cframe for the C-Band receiver, QWidget Lframe for the L-Band receiver, QWidget Pframe for the P-Band receiver, and QWidget LPframe fort hosting both QWidget Lframe and QWidget Pframe and all common QLabel objects.

Four slots have been reimplemented from SDSDevice virtual slots. Slot void SRT::enableDevice(bool bl) sets the value for member bool isEnabled and accordingly sets the graphic properties for SDSStatusButton deviceAct. Slot void SRT::readCustomLine(QString csl) is implemented for reading the setup lines that contain the setup keywords ActiveSurface and CalSource. Slot void SRT::stop(), which is meant to be used for leaving the antenna idle at a fixed position in azimuth and elevation, sends to NURAGHE its goTo command, by setting as arguments the current values for the antenna azimuth and elevation. Slot void SRT::start() checks first if some essential parameters have been set, then composes the string containing all necesary commands for configuring the antenna, by calling QString SDSAntenna::getReceiverConfigCommand(int rci), setting the pointing offsets and the coordinates to be tracked.

Eight slots have been reimplemented from SDSAntenna virtual slots. Slot void SRT::antennaPark() composes the command queue for parking the antenna. The given NURAGHE commands are antennaStop and telecopePark. Similarly slot void

SRT::antennaUnpark() composes the command queue for unparking the antenna. The given NURAGHE commands are antennaReset and antennaUnstow. Slot void SRT::checkAntennaStatus() is partially implemented, since its full implementation depends on the experiences in regularly using SRT Antenna Control. If the antenna is not enabled, in the main gui the values of the current source and its coordinates are displayed. It checks if the pointing status changes for storing it in the log file, it performs some checks whenever a change occurs in the antenna system status, checks if the antenna has reached either limit in azimuth and elevation, and emits the signals void SDSAntenna::currentPointingStatus(QString) and void SDSAntenna::currentSystemStatus(QString) for passing to other objects these informations.

Slot void SRT::getAntennaParameters(QString totString) reads the string returned by NURAGHE when its queried for the current antenna parameters, and stores the obtained values in the SDSEdit objects in auxiliaryWindow. Slot QString SRT::getReceiverConfigCommand(int ri) composes and returns the necessary command queue for configuring the selected receiver and accordingly the active surface shape and behaviour.

Slot void SRT::readReceiverLine(QString rcl) analyzes the setup line containing the keyword Receiver and accordingly shows the frame for the indicated receiver and sets the related parameters. Slot void SRT::selectReceiver(int i) (protected) shows the frame for the selected receiver, accordingly to the selected item in SDSCombo ReceiverCombo.

Slot void SRT::updateSourceParameters(QString str) reads the source string that is passed from the source catalogue and accordingly sets the source name and coordinates in members SDSEdit SourceName, SDSEdit Longitude and SDSEdit Latitude.

A new slot introduced in class SRT, QString SRT::getActiveSurfaceCommand(int), composes and returns the command queue for configuring the antenna's active surface shape and behaviour. Its argument is the index of the selected item in SDSCombo ActiveSurface.

Member *commThread has been set the string antennaPArameters as the device periodic command, and 2 s as the time interval between two consecutive commands.

Inheritance

Inherits from: SDSAntenna

Public members

`SDSCombo ActiveSurface`
This member is the combo box for selecting the active surface shape and behaviour.

`Protected members`

`QLabel CAttsLb`
This label provides the description for the `SDSCombo` objects devoted to set the attenuation levels for the `C-Band` receiver.

`SDSCombo Cbw`
This `SDSCombo` object allows the setting of the frequency band width to be observed with the `C-Band` receiver.

`QButtonGroup CcalBg`
This button group coordinates the two radio buttons `QRadioButton CcalOff` and `QRadioButton CcalOn` through which the calibration source is switched on and off for the `C-Band` receiver.

`QLabel CcalLb`
This label provides the description for the `QRadioButton` objects devoted to switch on and off the calibration source for the `C-Band` receiver.

`QRadioButton CcalOff`
This `QRadioButton` indicates, when selected, that the calibration source has to be switched off for the `C-Band` receiver.

`QRadioButton CcalOn`
This `QRadioButton` indicates, when selected, that the calibration source has to be switched on for the `C-Band` receiver.

QWidget Cframe

This QWidget is the frame that hosts all objects for setting the parameters for the C-Band receiver.

SDSEdit CLOfreq

This SDSEdit object is the field for setting the frequency of the local oscillator for the C-Band receiver. SDSCombo CSec0

This SDSCombo object allows to set the attenuation level for the signal's sector 0 of the C-Band receiver.

SDSCombo CSec1

This SDSCombo object allows to set the attenuation level for the signal's sector 1 of the C-Band receiver.

QWidget Lband

This QWidget is the frame that hosts all objects for setting the parameters for the L-Band receiver.

SDSCombo LbandsCb

This SDSCombo object allows to set the frequency band to be observed with the L-Band receiver.

SDSCombo LpolsCb

This SDSCombo object allows to set the polarization to be detected with the L-Band receiver.

QLabel LPAttsLb

This label provides the description for the SDSCombo objects devoted to set the attenuation levels for the L-P receiver.

QWidget LPframe

This QWidget is the widget for hosting the frames for the L-Band and P-Band receivers, and some QLabel objects of common use for both receivers' frames.

QLabel LPFreqsLb

This QLabel host the description for the SDSCombo objects that allow the selection of the

frequency bands to be observed with the L-Band and P-Band receivers.

SDSCombo LpolsCb
This SDSCombo object allows to set the polarization to be detected with the L-Band receiver.

QLabel LPPolsLb
This QLabel host the description for the SDSCombo objects that allow the selection of the polarizations with the L-Band and P-Band receivers.

SDSCombo LSec0
This SDSCombo object allows to set the attenuation level for the signal's sector 0 of the L-Band receiver.

SDSCombo LSec1
This SDSCombo object allows to set the attenuation level for the signal's sector 1 of the L-Band receiver.

QWidget Pband
This QWidget is the frame that hosts all objects for setting the parameters for the P-Band receiver.

SDSCombo PbandsCb
This SDSCombo object allows to set the frequency band to be observed with the P-Band receiver.

SDSCombo PpolsCb
This SDSCombo object allows to set the polarization to be detected with the P-Band receiver.

SDSCombo PSec0
This SDSCombo object allows to set the attenuation level for the signal's sector 0 of the P-Band receiver.

SDSCombo PSec1
This SDSCombo object allows to set the attenuation level for the signal's sector 1 of the P-Band receiver.

Public slots

`void SRT::antennaPark()`
This slot composes and sends to the communication thread `*commThread` the command queue for parking the antenna.

`void SRT::antennaUnpark()`
This slot composes and sends to the communication thread `*commThread` the command queue for unparking the antenna.

`void SRT::checkAntennaStatus()`
This slot performs all runtime checks on the antenna system and pointing status. Its implementation is partial since the regular use of `SRT Antenna Control` will highlight all peculiar cases when specific decisions have to be taken.

`void SRT::enableDevice(bool bl)`
This slot disables and enables the antenna control by setting the opportune boolean value in member `bool isEnabled` and accordingly changing the graphic properties of member `SDSStatusButton deviceAct`.

`void SRT::getAntennaParameters(QString parString)`
This slot reads the string obtained by sending to `NURAGHE` the command `antennaParameters`, and accordingly displays the obtained values in the `SDSEdit` objects hosted in `QWidget auxiliaryWindow`.

`QString SRT::getActiveSurfaceCommand(int ri)`
This slot composes and returns the command queue for setting the active surface shape and behaviour. Its argument is the index of the item set in member `SDSCombo ActiveSurface`.

`QString SRT::getReceiverConfigCommand(int ri)`
This slot composes and returns the command queue for setting the receiver and its parameters. Its argument is the index of the item set in member `SDSCombo ReceiverCombo`.

`void SRT::readCustomLine(QString cLine)`
This slot is implemented for reading the setup lines that contain the keywords `ActiveSurface` and `CalSource`.

`void SRT::readReceiverLine(QString) rLine`
This slot is implemented for reading the setup lines that contain the keyword `Receiver`.

`void SRT::start()`
This slot composes and sends to the communication thread `*commThread` the command queue for configuring the antenna and tracking the source's coordinates.

`void SRT::stop()`
This slot is meant for leaving the antenna idle at a fixed position in elevation and azimuth. To do so, it sends to the communication thread `*commThread` the `NURAGHE` command `goTo`, by giving it the current values for the antenna azimuth and elevation.

`void SRT::updateSourceParameters(QString sPar)`
This slot takes as argument the string passed by member `sourceCat`, whenever this latter is queried, and sets in members `SDSEdit SourceName`, `SDSEdit Longitude` and `SDSEdit Latitude` the values for the source name and its coordinates.

Protected slots

`void SRT::selectReceiver(int ri)`
This slot shows the frame that hosts all objects for setting the parameters for the selected receiver. Its argument is the index of the item selected in member `SDSCombo ReceiverCombo`.