D2C-GUI: A DATA CONVERTER FOR THE INAF RADIO TELESCOPES. I.

POSITION SWITCHING AND NODDING SPECTROSCOPIC OBSERVATIONS WITH THE SRT

F. Schirru[1], M. Bartolini[2], P. Castangia[1], S. Poppi[1], A. Tarchi[1], M. Bachetti[1], M. De Biaggi[3], A. Trois[1], S. Leurini[1], A. Orlati[3], A. Zanichelli[3]

[1] INAF - Osservatorio Astronomico di Cagliari, via della Scienza 5, 09047, Selargius (CA), Italy

[2] SKA Observatory, Jodrell Bank, Cheshire, UK

[3] INAF - Istituto di Radioastronomia, via Piero Gobetti, 101, 40129 Bologna (BO), Italy

Reviewers:
Dr Andrea Melis
Dr Alessandro Cabras

Anno 2025

Abstract

This technical report, intended as a first publication in a series of reports, introduces the `d2c-GUI` software, a tool developed to convert data collected by the Sardinia Radio Telescope (SRT) into a format suitable to be read out in CLASS (Continuum and Line Analysis Single-dish Software), part of the GILDAS (Grenoble Image and Line Data Analysis Software) package. After having introduced the motivations behind this work, some of the key concepts are presented along with a brief overview of the data file format of the Italian radio telescopes. Next, a description of the converter both in GUI (Graphical User Interface) and CLI (Command Line Interface) mode is provided. Eventually, a test session focused on data acquired with the SRT and the first results obtained by using the CLASS software are presented. A preliminary test, performed using spectroscopic measurements taken at the Medicina antenna, is also mentioned.

Contents

1 Motivation for developing a data format converter

Data coming out from the radio telescopes backends are commonly written in a standard FITS (Flexible Image Transport System) file format. The latter is typically designed to store, transmit, and manipulate scientific images and associated data.

According to the specific needs, FITS files may differ in their data organization, type or number of data columns, block sizes and header keywords used to carry ancillary information. For instance, SDFITS [3], based on standard FITS binary tables, is the available format at the Green Bank Telescope (GBT); RPFITS [4] is the format in use at the Australia Telescope National Facility (ATNF), in particular at Mopra and the Australian Telescope Compact Array (ATCA). The FITS format adopted by the Italian radio telescopes (Medicina, Noto and the SRT) [2] is organized in extension tables and it is similar to the SDFITS. Due to the existence of various FITS conventions, a unique software able to read out and analyze output data is not available. For this reason, each radio astronomical facility makes use of one or more of the following solutions:

- *proprietary software*

  One or more researches and/or technologists usually (but not exclusively) belonging to the Institute running the observational facility, on a voluntary basis, develop and maintain the software together with possible documentation. It normally requires an authorization by the developer(s) for its usage and it needs to be learned by new users.

- *public software*

  By definition, it is made available to anybody and it is provided with some documentation. It can be downloaded and upgraded by developers according to their needs. Its maintenance is not guaranteed and it needs to be learned by the observers at the first instance of its usage. In this category are included also packages like AIPS (Astronomical Image Processing System) [5], GILDAS (Grenoble Image and Line Data Analysis Software) [6], CASA (Common Astronomy Software Applications) [7] and MIRIAD (Multichannel Image Reconstruction, Image Analysis and Display) [8] that have often been initially developed to meet the needs of specific telescopes and rapidly became a reference for astronomical data analysis.

- *station's software*

  It is developed and maintained by the observational station upon an endorsement by the institution running the research facility. The software is customized to meet specific needs and is made available to the observers at the facility, complemented by detailed documentation.

  A station's software aiming at performing data reduction and analysis requires a broad range of data processing capabilities, including data flagging, imaging and cleaning and therefore requires significant effort in terms of manpower.

  A cost-effective, complementary approach to data reduction for an observing facility can be the development of format converter(s) to allow for compatibility and exploitation of the available public data reduction software. Regardless of the FITS file format adopted, raw data recorded by any radio telescope can be converted and loaded into a public software which is well known, maintained and provided of documentation (see also previous item). The use of a data converter, which all in all is a station's tool,

certainly has the advantage of increasing the number of users of any radio telescope. However, depending on the actions performed by the converter on the raw data (i.e. simple reordering and/or calculations), it is crucial to perform tests to verify the quality of the output data.

In this report we present the `d2c-GUI` data converter, that has to be intended as a station's software, an application written in Python and based on an upgraded version of the `discos2class` tool which was developed by M. Bartolini in the 2016 [11]. It allows to convert data recorded by the SRT (and, potentially, Medicina and Noto; see also Sect. 7) into a format suitable to CLASS, one of the software packages which belongs to GILDAS.

2 Scan, subscan and duty cycle

With some differences, the three Italian single dish radio telescopes support standard observing modes for spectroscopic observations, including *position switching*, *nodding* and *on-the-fly mapping*. In the following we will focus on the position switching and nodding strategies which correspond to an on-source/off-source pointing sequence called *scan*. A scan is, in turn, divided into *subscans*, each of which represented by a recorded FITS file. The off-source spectrum is used to remove atmospheric and system noise, and to bandpass calibrate the on-source spectrum.

According to the observing mode and strategy, scans have a specific pattern called *duty cycle*[1]. In addition, the number of feeds used to record data differs: it is one for the position switching mode and two for the nodding mode.

2.1 Duty cycle in position switching mode

A schematics of a duty cycle configuration relative to the position switching mode is shown in Figure 1-[A][2]. Its pattern can be written as

`<on>:<off>:<off_cal>`

where:

- `<on>` represents the number of subscans with the feed pointing the source;
- `<off>` is the number of subscans with the feed off source;
- `<off_cal>` represents the number of subscans with the feed off source and the calibration mark switched on.

2.2 Duty cycle in nodding mode

Figure 1-[B] reports a schematics of a the duty cycle configuration relative to the nodding mode [14]. In this case, it can be written as

`<a_cal>:<a>:<b>:<b_cal>`

[1] A scan can include multiple duty-cycles.

[2] It is an example among all possible configurations.

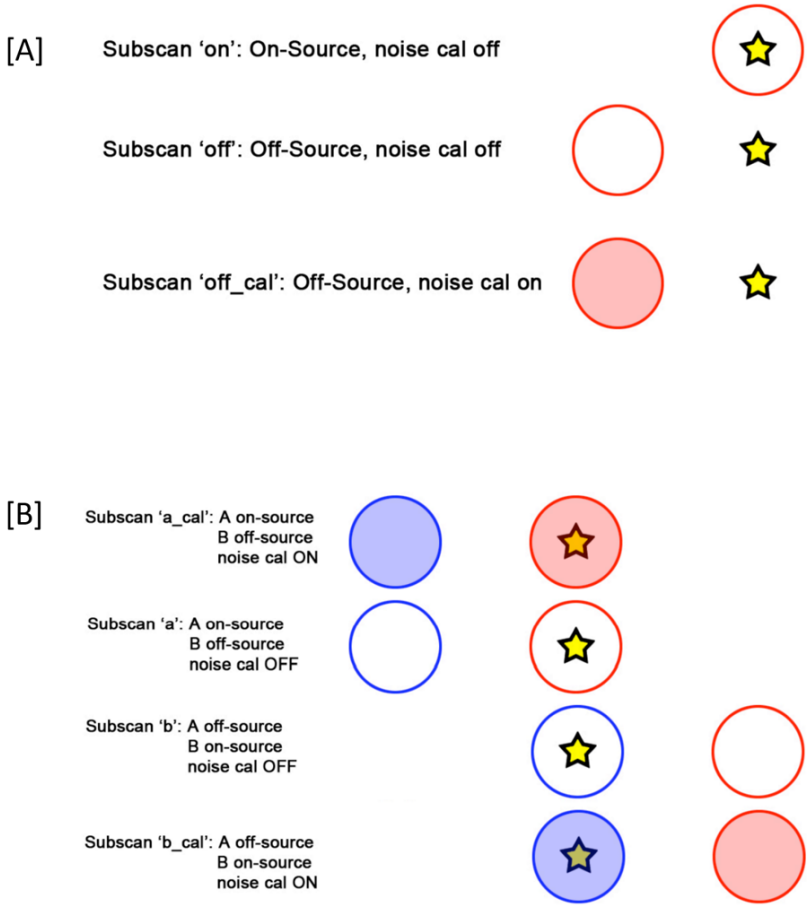

Figure 1: Duty cycle schematics relative to the position switching [A] and nodding [B] mode. Data are recorded with one or two feeds respectively [9].

where:

- `<a_cal>` represents the number of subscans with the primary feed pointing the source, the secondary feed off source and the calibration mark switched on;
- `<a>` is the number of subscans with the primary feed pointing the source and the secondary feed off source;
- `<b>` is the number of subscans with the secondary feed pointing the source and the primary feed off source;
- `<b_cal>` represents the number of subscans with the secondary feed pointing the source, the primary feed off source and the calibration mark switched on.

3 The standard FITS format at the INAF radio telescopes

During an observation, each subscan composing the duty cycle pattern is stored into a standardized FITS format [2]. It consists in one or more *Header* and *Data Units* (HDUs) where the first HDU is called the *Primary*

HDU, or *Primary* array while any additional HDUs which may follow the Primary array is called *extension*. Extensions can be:

- **Image**: an N-dimensional array of pixels.
- **Binary Tables**: rows and columns of data in binary representation.

Figure 2 shows the structure of a FITS file produced at the SRT by using SARDARA, one of the backends available at the telescope and fully integrated in the control system [12].

The SECTION TABLE extension contains, among the others, information about the data type, the number of channels contained in each recorded spectrum (for instance 1024, 4096, 16384), relevant frequencies and bandwidth of observation. Its rows correspond to the data columns in the DATA TABLE extension, each of which containing a certain number of recorded spectra.

The data type reported in the SECTION TABLE extension assumes three values: *simple*, *spectra* and *stokes*. They are referred to data from total power, dual-polarization (LL and RR) and full polarization (LL, RR, LR and RL) spectroscopic observations, respectively. According to the data type, the number of rows of the SECTION TABLE extension differs. In particular:

- data type spectra produces two rows per feed, that is, data relative to the polarization LL and RR are separated in two different DATA TABLE columns.
- data type stokes generates one row per feed, being all data relative to the polarization LL, RR, LR, RL merged in one single DATA TABLE column.

Details about the feeds used to record the data can be extracted from the FEED TABLE extension; information about the observation, the source, the number of scan and subscan can be deduced from the keywords stored in the Primary extension. In particular, the keyword SIGNAL takes care of the position of the feed respect to the source under observation and it is used to reconstruct the duty cycle pattern.

Table 1 shows the values assumed by the keyword SIGNAL and their correspondence with the duty cycle pattern.

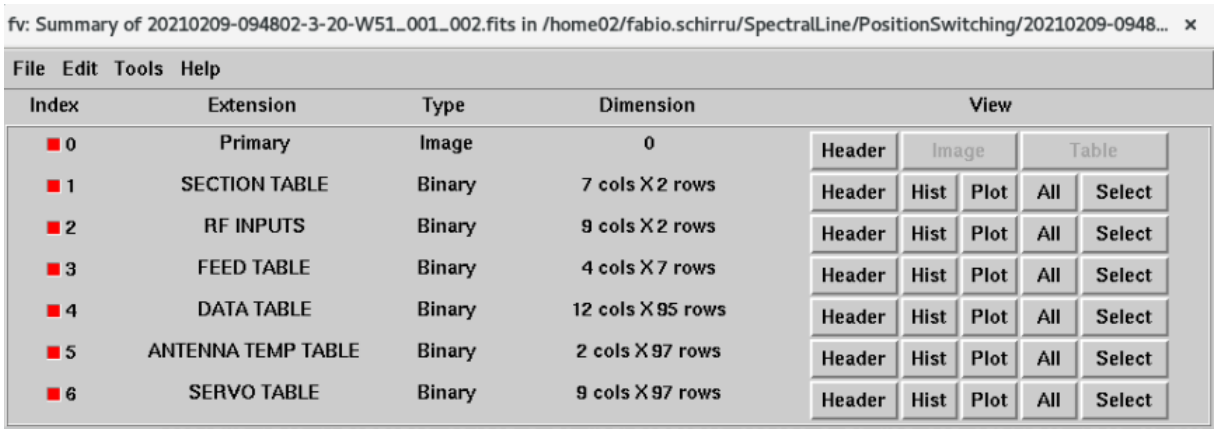

Figure 2: Structure of a FITS file as presented by the FITS viewer *fv* [10].

SIGNAL Value	Duty Cycle Value	Mode
REFSIG	a_cal	Nodding
SIGNAL	on, a	Position Switching, Nodding
REFERENCE	off, b	Position Switching, Nodding
REFCAL	off_cal, b_cal	Position Switching, Nodding

Table 1: Values of the keyword SIGNAL and their correspondence with the duty cycle pattern.

3.1 Summary FITS file

Each scan is characterized by a summary file, identified by the prefix `"Sum_"`, listing the metadata relevant to describe the observation, for archival search and data processing. In particular it contains metadata values requested by the converter (see Sect. 4) such as:

- *RESTFREQ*: that is the value of signal frequency as seen from the source. The parameter is given per feed and per polarization.
- *VRAD*: that is the radial velocity, the component of the source velocity that is directed along the line of sight of the radio telescope.

4 The converter

The developed converter is available in GUI and CLI modes and it is based on an upgraded version of the `discos2class` tool [11]. The latter is made up of three Python modules:

- `__init__.py`: it receives the user parameters provided by the `discos2class` command and starts the data processing by calling the `discosscan.py` module.
- `discosscan.py`: it calls the following functions:
  - `load_subscans()`: for each scan, it generates a list of subscans to be processed, ordered by their recording time stamp;
  - `load_summary_info()`: as reported in Sect. 3.1 it loads the *RESTFREQ* and the *VRAD* parameters from the `Sum_*.fits` file.
  - `convert_subscans()`: it calls the module `scancycle.py` and saves to disk a `.gdf` file containing the data converted into the CLASS format.
- `scancycle.py`: for each subscan belonging to a duty-cycle within the scan, at first, it extracts the value of the keywords from the header, the polarization type, the raw data, their length[3] and the integration time. Next, for each value of the keyword SIGNAL and polarization type, extracted raw data are progressively added into a "Python dictionary" and eventually averaged at the end of each scan. These values are then used to compute the averaged spectrum relative to each scan as (`<on>.mean` - `<off>.mean`) / `<off>.mean`.

[3]The data length corresponds to the number of spectra included in each subscan. As reported in Sect. 3 this also is the number of rows of the data columns located in the DATA TABLE extension.

4.1 GUI mode

According to the schematics reported in Figure 3, the `d2c-GUI` converter generates a string of parameters which is used to build the `discos2class` command. Parameters are read out by the `__init__.py` module of the the `discos2class` tool which, in turn, generates a `.gdf` file containing the data converted into the CLASS format. `d2c-GUI` includes the following modules:

- `d2c-gui.ui`: it is an XML file created by using the *Qt Designer* and contains the GUI components of the converter.
- `d2c-gui_loader.py`: it is a Python file which loads and initializes the `d2c-gui.ui` file. This is accomplished by using the *uic.loadUI()* method included in the *uic* module of the *PyQt5* library which allows to make the *d2c-gui.ui* file a fully-functional PyQt5 object. The Python file includes also methods which give fully functionality to the GUI components.

Figure 4 shows the graphical user interface of the converter. It includes a control panel and a log viewer area. The former is divided in *Data Source*, *Duty Cycle* and *Data Processing* sections while the latter displays the `discos2class` command to be executed and information related to the results of the data validation and conversion.

The *Data Source* section includes the following components:

- **BACKEND** [combo box]: it allows to select the backend which was used to record the data. Options available are *SARDARA_BKD* and *SKARAB_BKD*, that are the two backends typically used for spectroscopic observations, namely SARDARA [12] and SKARAB [13]. Each option connects the converter to a specific storage area containing the acquired raw data. In case the connection fails, all components of the *Data Source* section get disabled and the conversion can not be performed.
- **UPDATE** [button]: it allows to reload the drive folder structure containing the raw data without forcing the user to close and open again the application.
- **PROJECT ID** [combo box]: it shows the project id. The latter corresponds to the username currently authenticated on the machine which runs the converter. The project id represents the root of the data folder structure.

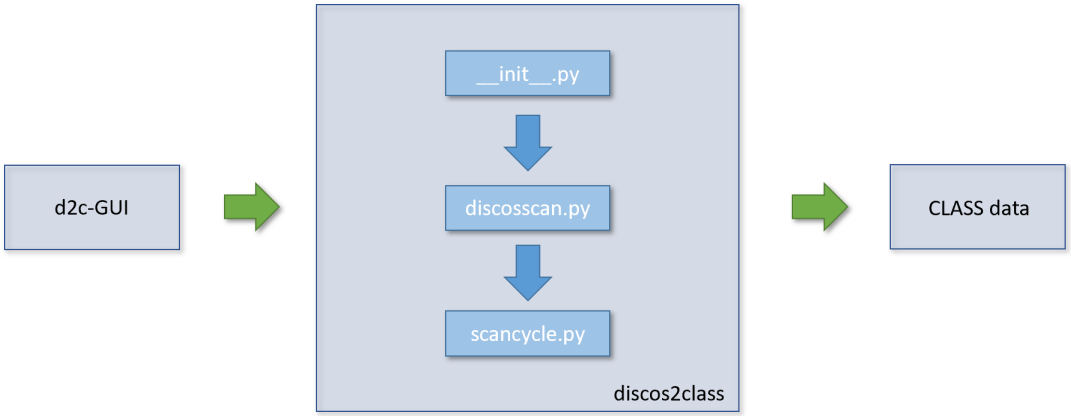

Figure 3: Schematics of the working principle of the converter in GUI mode.

Figure 4: Graphical user interface of the `d2c-GUI` converter.

- **DATE** [combo box]: it allows to select datasets recorded at a specified date relative to the selected project id. The component shows only folders whose names match the '`yyyyMMdd`' date format. These folders represents the first level in the data folder structure.
- **DATASET** [combo box]: it allows to select a dataset relative to an observation performed at the specified date. In this case, the folder represents the second and last level in the data folder structure. The component displays only dataset folders which contain the summary FITS file (`Sum_*.fits`) which is mandatory for the data conversion. In case no datasets are found within the selected date, the component displays *UNAVAILABLE* and the conversion can not be performed.
- **ALL FOLDERS** [check box]: when checked, it allows the user to convert all folders contained within the selected date. If the component remains unchecked, only the dataset specified in the DATASET combo box will be converted.
- **DESTINATION** [button]: it allows the user to specify the folder which will store the converted `.gdf` files.

The *Duty Cycle* section includes the following components:

- **MODE** [combo box]: it allows to select the observation mode as *Position Switching* or *Nodding*.

- **PATTERN** [combo box]: it allows the user to set the duty cycle configuration[4]. Each index can be selected via combo box, with values ranging from 0 to 9, where the first and last values are reserved for calibration purposes (see Sect. 2.1 and 2.2 for a complete description of the duty cycle pattern indexes).

- **CHECK** [button]: it starts the dataset validation. The latter is performed by checking whether the order of the subscans of each scan matches the duty cycle pattern. This is achieved by reading out the value of the SIGNAL keyword from each individual subscan and comparing it with that expected from the duty cycle pattern. The status of the dataset validation is displayed by the progress bar located at the left side of the component.

  If the validation terminates with no errors, the START button located in the the *Data Processing* section gets enabled and the data conversion is ready to be performed.

  In case of errors (wrong SIGNAL keyword, missing or corrupted data file) the validation procedure will display a message in the log viewer area indicating at which duty cycle within the scan and data file the error was encountered.

  The CHECK button gets enabled only by selecting a dataset from the *Data Source* area and setting the duty cycle pattern with a size (sum of each index value) greater than zero. The button gets disabled any time a change in the GUI parameters is detected.

The *Data Processing* section includes the following components:

- **SKIP CALIBRATION** [check box]: when checked, it allows to skip the counts-to-kelvin calibration of the entire dataset which has to be converted. In this case, the resulting intensity of the spectrum obtained from each individual duty cycle within the scan is simply computed as (`<ON>.mean - <OFF>.mean`) / `<OFF>.mean`.

  The counts-to-kelvin calibration is performed by using the calibration mark temperature (`<Tcal>`) which is extracted from the metadata of the original FITS files. As an example, considering a scan recorded in position switching mode, for each duty-cycle, the procedure would run as follows:

  → `<Tcal> / (<off_cal>.mean - <off>.mean) = counts2kelvin`

  → `Tsys = counts2kelvin * <off>.mean`

  → `SpectrumInKelvin = (<on>.mean - <off>.mean) * counts2kelvin = = (<on>.mean - <off>.mean) * Tsys / <off>.mean`

  Regardless of the observation mode, the counts-to-kelvin calibration is performed only for any duty-cycle within the scan containing, at least, one `<_cal>`-type subscan.

- **START** [button]: according to the user inputs, it generates the `discos2class` command and executes it. The status of the dataset conversion is displayed by the progress bar located at the left side of the component.

  If a change in the GUI parameters is detected, the button gets disabled and the validation procedure must be performed again.

[4]Three or four entries are enabled for the Position Switching and Nodding modes respectively. See Table 1 for their corresponding values.

4.2 CLI mode

The d2c-GUI converter can be launched directly from command line by typing the discos2class command. As reported for the GUI mode, it is possible to convert scans from multiple source folders. The syntax to be used is as follows[5]:

```
discos2class [-h] [-d] [-o OUTPUT_DIR] [-c DUTY_CYCLE] [-s] [-version]
             source_dir [source_dir ...]
```

Argument	Type	Description
source_dir	positional	Directory path(s) to scans
-h, -help	optional	Shows the help message and exist.
-d	optional	Enables debug messages
-o OUTPUT_DIR	optional	Output directory name.
-c DUTY_CYCLE	optional	Duty cycle pattern of each scan as <on>:<off>:<cal> or <a_cal>:<a>:<b>:<b_cal>. Elements must be all presents but can be zeroes.
-s	optional	Skips the counts-to-kelvin calibration and computes the quantity (<ON>.mean - <OFF>.mean) / <OFF>.mean by ignoring the <_cal>-type subscans.
-version	optional	Prints version information and exits

Table 2: Arguments of the discos2class command and their description.

As an example, suppose to have:

- Source folder: /20240821-134018-2623-IC485
- Destination folder: /converter_data/results
- Duty cycle: <1>:<6>:<6>:<1> (Nodding mode)
- Debug mode: True
- Skip calibration: True

The discos2class command to be executed will be the following:

```
discos2class -d -o /converter_data/results -c 1:6:6:1 -s
             /20240821/20240821-134018-2623-IC485
```

To be noted that in case the chosen destination folder does not exist, it gets automatically created.

[5]Refer to Table 2 for the description of each argument.

5 Test session [GUI mode]

A conversion test was carried out on a dataset relative to a nodding project labeled 26-23, performed on IC 485, with a 1:6:6:1 duty cycle. The procedure to convert the dataset into a `.gdf` CLASS is described in the following:

- Set BACKEND to "SARDARA_BKD". This allows the converter to directly access (read-only mode) the specific storage area containing the raw data recorded with the SARDARA backend. Therefore, it is not necessary to make copies or move the data to other locations.
- Check that the PROJECT ID combo box displays the code "26-23". This corresponds to the username logged on the machine which runs the converter.
- Set DATE to "20240821".
- Set DATASET to "20240821-134018-2623IC485".
- Set DESTINATION by pressing the button SELECT. This will open a folder browser dialogue. Next, navigate and select `/converter_data/results`.
- Set MODE to "Nodding".
- Set the duty Cycle pattern to `<1>:<6>:<6>:<1>`. The CHECK button should get enabled.
- Press the CHECK button to start the dataset validation. If the procedure terminates successfully, the "START" button gets enabled. Otherwise, an error message will be displayed in the log viewer area.
- Press the START button to convert the dataset. The `discos2scan` command will be displayed in the log viewer area and executed. If the process terminates successfully, the selected destination folder now contains a CLASS file named `20240821-134018-2623-IC485_nod.gdf` which holds the result of the integration and calibration of the duty-cycles within the scan. So far, the time required for the data conversion in this test (8.75 GB of disk usage) is 2 minutes and 20 seconds.

  It has to be noted that the converter must be launched once the schedule has been closed. In fact, the `Sum_*.fits` file (see Sect. 3.1), from which keywords (needed by the converter) are extracted, is presently finalized at the very end of the schedule. Therefore, the shorter the scan (containing fewer duty-cycles), the more promptly the conversion can be performed.

Next, to inspect the converted file with CLASS, the following commands (which are only illustrative) can be provided:

- Launch the software by typing into the terminal the command '`class`'.
- Type '`file in 20240821-134018-2623-IC485_nod.gdf`' to open the converted data.
- Type '`find/all`' to get the list of spectra per feed and per polarization contained in the dataset. The list can be filtered out by using the commands '`set Line`' and '`set Telescope`'. For instance, the command '`set Line F0-420.0`' followed by '`find`', filters out all data relative to the feed 1 (F1-420.0). To go back to the original dataset, type the command '`set Line *`' followed by '`find`'.

- Type '`list`' to display the list of spectra (see Figure 5).
- Type '`set unit v c`' to set the top and bottom x axis units (in this case, v = velocity, c = channels).
- Type '`set mode x 8200 8500`' to set the x axis range.
- Type '`set bad and`' to prevent CLASS from generating a plot with empty bins (the software interprets a channel with amplitude 0 as a bad channel).
- Type '`average /nocheck`' to compute an average of the entire selected dataset, ignoring the differences in the feed used to acquire the data.
- Type '`plot`' to display the converted dataset (see Figure 6).

```
LAS> file in  2024234_IC485_nod.gdf
I-INPUT,  2024234_IC485_nod.gdf successfully opened
LAS> find /all
I-FIND,  100 observations found
LAS> list
Current index contains:
  N;V Source       Line         Telescope      Lambda     Beta Sy Sca Sub
  1;1 IC485        F0-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1   2
  2;1 IC485        F0-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1   2
  3;1 IC485        F1-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1   2
  4;1 IC485        F1-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1   2
  5;1 IC485        F0-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1  16
  6;1 IC485        F0-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1  16
  7;1 IC485        F1-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1  16
  8;1 IC485        F1-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1  16
  9;1 IC485        F0-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1  30
 10;1 IC485        F0-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1  30
 11;1 IC485        F1-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1  30
 12;1 IC485        F1-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1  30
 13;1 IC485        F0-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1  44
      :                              :                                :
 87;1 IC485        F1-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1 296
 88;1 IC485        F1-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1 296
 89;1 IC485        F0-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1 310
 90;1 IC485        F0-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1 310
 91;1 IC485        F1-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1 310
 92;1 IC485        F1-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1 310
 93;1 IC485        F0-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1 324
 94;1 IC485        F0-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1 324
 95;1 IC485        F1-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1 324
 96;1 IC485        F1-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1 324
 97;1 IC485        F0-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1 338
 98;1 IC485        F0-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1 338
 99;1 IC485        F1-420.0     SRT-K-SAR-LL     +0.0     +0.0 Eq   1 338
100;1 IC485        F1-420.0     SRT-K-SAR-RR     +0.0     +0.0 Eq   1 338
```

Figure 5: Content of the `.gdf` CLASS file. Each line corresponds to a specific feed number and polarization type for each duty-cycle. The column "Line" includes the information relative to the feed number and the bandwidth used; the radio telescope name, the frequency band, the backend used to record data and the polarization type are instead indicated in the "Telescope" column.

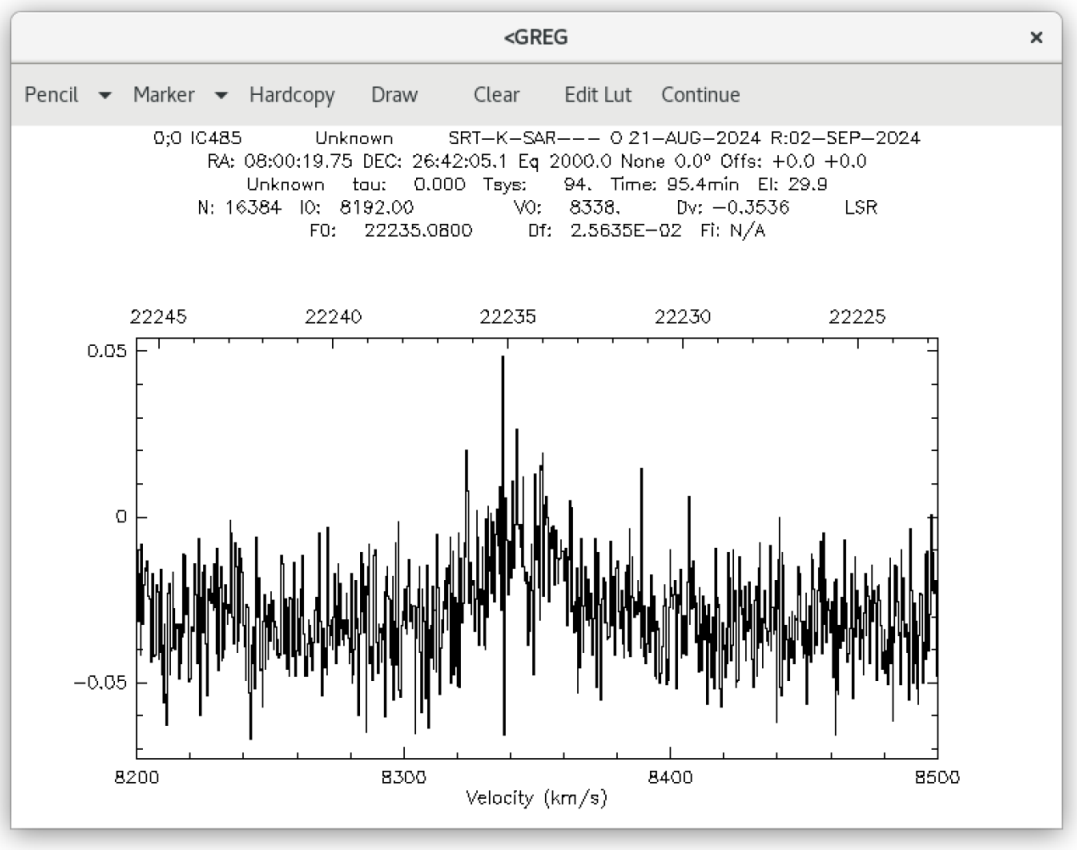

Figure 6: Spectrum of the water megamaser emission in the LINER AGN galaxy IC 485. The plot is obtained from the CLASS file `20240821-134018-2623-IC485_nod.gdf` by averaging over the entire dataset recorded during the observations. The raw data are taken, in nodding mode, at the SRT with the SARDARA backend in the framework of the project #26-23 (courtesy E. Ladu).

6 Requirements and Installation

The `d2c-GUI` data converter is developed in Python 3.11.8. It has been tested with the versions of the packages reported in Table 3. The version of the GILDAS software (in our case, the *apr24a* release) to be used is the one equipped with the Pyclassfiller Python package.

Package Name	Version
astropy	6.1.0
matplotlib	3.8.4
pyqt5-tools	5.15.9.3.3
pyclassfiller	apr24a

Table 3: List of the required Python packages.

The software is available at `https://github.com/discos/discos2class/tree/development` and can be installed with the following command:

- `python setup.py install`

To launch the converter, type the command:

- `python d2c_gui_loader_v2_0.py`

7 Final Remarks

The converter described in this report is aimed at handling data produced by spectroscopic observations taken with the Italian radio telescopes.

Presently, it has been thoroughly tested on data acquired with the SRT in position switching and nodding modes.

A preliminary test has also been successfully performed using spectroscopic measurements taken in position switching mode at the Medicina antenna, with the XARCOS backend [15]. An illustrative spectrum is shown in Fig. 7. The details of the converter upgrade to handle Medicina (and Noto) data will be, however, described in a forthcoming publication of this series of reports.

In addition, future developments of the `d2c_gui` converter foresee, among others, to extend its capabilities in order to deal also with data taken using other observing techniques like, e.g., raster scan and on-the-fly mapping. Eventually, the software, together with a user manual, will be made available at the SRT (and, ultimately, at the other Italian antennas), thus maximizing a prompt exploitation of it during the observations.

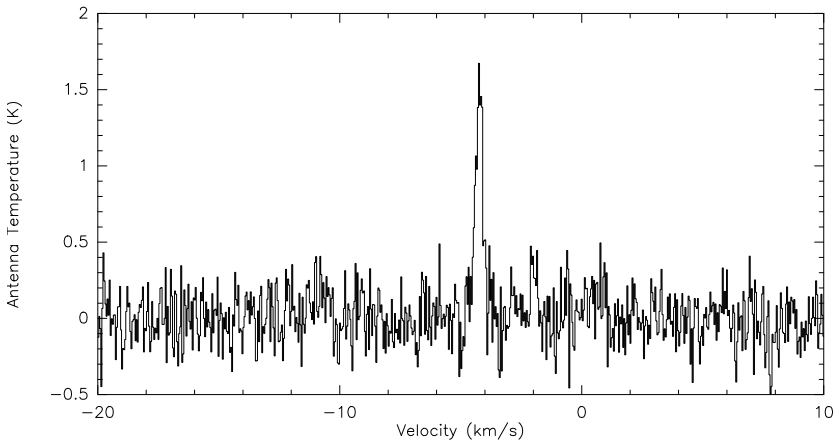

Figure 7: Spectrum of the methanol maser in the Galactic star forming region DR 20 source taken at Medicina with the XARCOS backend. The raw data were taken in the framework of the project #22-23 (courtesy J. Brand) and have been preliminary converted with the `d2c_gui` converter (see Sect. 7 for details).

Acknowledgement

We would like to thank E. Ladu and J. Brand for providing the observational data and relevant feedback on IC 485 (SRT project: #26-23) and DR 20 (Medicina project #22-23), respectively. FS and AT are grateful to M. Murgia for fruitful discussions.

References

[1] I. Prandoni et al., *The Sardinia Radio Telescope . From a technological project to a radio observatory*, ASTRONOMY & ASTROPHYSICS, 608, A40, (2017)

[2] S. Righini, A. Zanichelli, *FITS output format in DISCOS*, SRT-MAN-10000-003, Issue n. 4, (2018)

[3] `https://casa.nrao.edu/aips2_docs/notes/236/node14.html`

[4] `https://https://www.atnf.csiro.au/computing/software/rpfits.html`

[5] `https://casa.nrao.edu/aips2_docs/gettingstarted.html`

[6] `https://www.iram.fr/IRAMFR/GILDAS/`

[7] `https://casa.nrao.edu/`

[8] `https://www.atnf.csiro.au/computing/software/miriad/`

[9] M. Bartolini, S. Righini, *basie User Manual*, IRA Technical Report, IRA 492-16 (2016)

[10] `https://heasarc.gsfc.nasa.gov/docs/software/ftools/fv/`

[11] `https://github.com/discos/discos2class`

[12] A. Melis et al., *SArdinia Roach2-based Digital Architecture for Radio Astronomy*, Journal of Astronomical Instrumentation, Vol. 07, No. 01, 1850004 (2018)

[13] A. Melis et al., *The SKARAB Board in the Framework of Single-Dish Radio Astronomy*, Journal of Astronomical Instrumentation, Vol. 13, No. 2, 2450008 (2024)

[14] A. Tarchi et al., *Spectroscopic observations with the SRT using the NODDING technique*, OACA Internal Report N. 72, (2018)

[15] A. Melis et al., *Integration of the digital full-Stokes spectrometer XARCOS into the control software of Sardinia & Medicina radio telescopes*, OACA Internal Report N. 52, (2015)