
Control System for the IRAM 30m Online Data Processing under Unix

Title: Online Data Processing

Identifier—MasterURL:

<http://www.iram.es/IRAMES/documents/projectDataProcOnline/dataProcOnline.xml>

File:

Revision: draft, 1.0

Revision Date: 2001-11-15

Expiration Date:

Supersedes: not applicable

Is superseded by: not applicable

Authors: W. Brunswig (brunswig@iram.es)

Contributors: H. Ungerechts (ungerechts@iram.es)

Affiliations:

Addresses:

Audience: technical staff, astronomers

Publisher: IRAM, Granada, Spain

Subject: Keywords: data processing, data format, raw data, calibration

Keywords: NCS

Description - about this document:

This projects will implement online data processing under Unix. It will be one of the main features of the new control system but we foresee to install part of it already also in the current control system. This version of the document concentrates on the installation in the current control system.

Note the list of pending items.

References:

See the list in Appendix [B](#).

References and Related Documents—Short List:

1. [pipeline concept](#)
2. [project rawDataToUnix documentation](#)
3. [Project Abba Control](#)
4. [Project Hera Control](#)

Pending Items:

1. check abbaToFits, heraToFits: the names of the pipe directories have changed!
2. end of scan is not always detected correctly
3. add description on what shall be done for each observing mode
4. port abbaToFits and heraToFits to Linux

Contents

1	Project Plan (2001-11-15)	5
2	Introduction (2001-11-15)	5
3	Requirements	5
3.1	* dataOnlineProcessing (<i>state: fixed 2001-11-08</i>)	5
3.2	* dataTransferIramEA (<i>state: done 2001-11-07</i>)	5
3.3	* dataTransferOthers (<i>state: done 2001-11-07</i>)	5
3.4	* processCalibration (<i>state: fixed 2001-11-08</i>)	6
3.5	* processSubscans (<i>state: fixed 2001-11-08</i>)	6
3.6	* processScan (<i>state: fixed 2001-11-08</i>)	6
3.7	* processCancelledScans (<i>state: fixed 2001-11-08</i>)	6
3.8	* feedbackObs (<i>state: fixed 2001-11-07</i>)	6
3.9	* logging (<i>state: fixed 2001-11-07</i>)	6
3.10	* plots (<i>state: fixed 2001-11-07</i>)	6
3.11	* processPointing (<i>state: open 2001-11-08</i>)	7
3.12	* processFocus (<i>state: open 2001-11-08</i>)	7
3.13	* processSkidip (<i>state: open 2001-11-08</i>)	7
3.14	* processPositionSW (<i>state: open 2001-11-08</i>)	7
3.15	* processFrequencySW (<i>state: open 2001-11-08</i>)	7
3.16	* processWobblerSW (<i>state: open 2001-11-08</i>)	7
3.17	* processOTF (<i>state: open 2001-11-15</i>)	7
3.18	* abbaToFits (<i>state: done 2001-11-15</i>)	9
3.19	* heraToFits (<i>state: done 2001-11-15</i>)	9
3.20	* processWhatElse (<i>state: open 2001-11-08</i>)	9
4	Specifications	9
4.1	* obsProcedure (<i>state: done 2001-11-07</i>)	9
4.2	* endOfScan (<i>state: fixed 2001-11-08</i>)	9
4.3	* workingDirectory (<i>state: fixed 2001-11-09</i>)	9
4.4	* scriptCoding (<i>state: fixed 2001-11-09</i>)	9
4.5	* taskControl (<i>state: fixed 2001-11-09</i>)	9
4.6	* taskCrashes (<i>state: fixed 2001-11-09</i>)	10
5	Design	10
5.1	* architectureOverview (<i>state: done 2001-11-13</i>)	10
5.2	* obsProcedure (<i>state: done 2001-11-07</i>)	10
5.3	* dataTransferIramEA (<i>state: done 2001-11-07</i>)	10
5.4	* dataTransferOthers (<i>state: done 2001-11-07</i>)	11
5.5	* endOfScan (<i>state: open 2001-11-08</i>)	11
5.6	* processCancelledScans (<i>state: open 2001-11-08</i>)	11
5.7	* processingPipeline (<i>state: open 2001-11-13</i>)	11

5.8	* calibration (<i>state: open 2001-11-13</i>)	12
5.9	* feedbackObs (<i>state: open 2001-11-07</i>)	13
5.10	* monitoring (<i>state: open 2001-11-07</i>)	13
5.11	* processCalibration (<i>state: open 2001-11-08</i>)	13
5.12	* processOTF (<i>state: open 2001-11-15</i>)	13
5.13	* abbaToFits (<i>state: done 2001-11-15</i>)	14
5.14	* heraToFits (<i>state: done 2001-11-15</i>)	14
6	Implementation	14
6.1	* obsProcedure (<i>state: done 2001-11-07</i>)	14
6.2	* dataTransferOthers (<i>state: done 2001-11-07</i>)	15
6.3	* processCalibration (<i>state: open 2001-11-07</i>)	15
6.4	* dataProcessingAbbaToFits (<i>state: open 2001-11-07</i>)	15
6.5	* dataProcessingHeraToFits (<i>state: open 2001-11-07</i>)	15
6.6	* processOTF (<i>state: open 2001-11-15</i>)	15
6.7	* abbaToFits (<i>state: done 2001-11-15</i>)	17
6.8	* heraToFits (<i>state: done 2001-11-15</i>)	17
7	Installation	17
7.1	* obsProcedure (<i>state: done 2001-11-07</i>)	17
7.2	* processCalibration (<i>state: open 2001-11-07</i>)	18
8	Logbook	18
8.1	* meet2001-11-12 (<i>state: done 2001-11-13</i>)	18
A	List of Requirements/Specifications and Descendants	19
B	References	21

1 Project Plan (2001-11-15)

1. 2001-11 version 1.0: Project documentation, basis of installation in the current control system, fix requirements, first ideas about design
2. 2001-12: design architecture
3. 2001-12: design of the script to process OTF Maps and calibrate data
4. 2002-02: first installation should be operational

2 Introduction (2001-11-15)

In the current MRT control system, antenna control is done on a slow Vax/VMS system (iramea). For some backends, the data is recorded on iramea and automatically processed at the end of subscans and scans (pointing fits, focus, skydips, calibrations, ...) by RED . However, due to the limited processing power of the antenna control computer it is desirable to do all the automatic data processing on a fast Unix workstation. In the future, we foresee that all backends will send their data directly to the MRT file server.

The new control system of the 30m RT foresees the implementation of a subsystem data processing. Any development of the current control system shall also take the future system architecture into consideration and can serve as a prototype for the design of the new control system.

3 Requirements

3.1 * dataOnlineProcessing (*state: fixed 2001-11-08*)

The data recorded by the 30mRT shall be processed under Unix directly after the end of subscans and scans. The processing done depends on the observing procedure of the current scan and the receivers and backends used.

3.2 * dataTransferIramEA (*state: done 2001-11-07*)

All header and raw-data files produced by the antenna control software shall be transferred automatically to the MRT file server (curr. mrt-ux1) at the end of each subscan.

3.3 * dataTransferOthers (*state: done 2001-11-07*)

Some backends send data directly to the MRT file server: 4MHz, ABBA. This data has to be merged with the data files from the antenna control system.

3.4 * processCalibration (*state: fixed 2001-11-08*)

Data related to the heterodyne receivers shall be calibrated online under Unix. The data processing software shall write the calibrated data (CLASS format) in a file in a directory of the project.

3.5 * processSubscans (*state: fixed 2001-11-08*)

For most observing modes it is required that results (normally in form of plots) are displayed at the end of subscans. (This is also called "quick look".)

3.6 * processScan (*state: fixed 2001-11-08*)

At the end of a scan, the data processing is different from the "quick look" processing.

3.7 * processCancelledScans (*state: fixed 2001-11-08*)

Data from cancelled scans shall be processed as well as far as reasonable, e.g. a pointing cancelled after subscan 2 could provide already the azimuth pointing error.

Note: *this feature will not be available in the 1st version.*

3.8 * feedbackObs (*state: fixed 2001-11-07*)

The data processing software has to send results (pointing offsets) back to the OBS program on iramea.

Note: *this feature will not be available in the 1st version.*

3.9 * logging (*state: fixed 2001-11-07*)

The data processing software shall write results into general log files (or into data bases in the future): calibration.log, pointing.log, (?? what else ?? please let me know). The observer shall have a window where this information is displayed.

Note: *In the NCS, this logging information will be send to the general monitoring process. It shall be checked if this concept can also be used in the current system. We also have to know, which software is accessing the current log files.*

3.10 * plots (*state: fixed 2001-11-07*)

Result from the online processing can result in plots, e.g. pointing, focus, skydip, spectra (on-off). These plots shall be displayed to the observer (also remote). The plots shall be written into files to displayed later. Information about the generated plots shall be kept also to allow for fast access (browsing) of the plots of a project. The plots shall be kept in a project directory.

3.11 * processPointing (*state: open 2001-11-08*)

Note: *For each observing procedure we should write down the requirements. This requirement is just an incomplete example.*

The processing of pointing scans shall:

1. at the end of each subscan display the recorded data, including a gauss fit
2. at the end of a scan determine the pointing corrections for azimuth and elevation
3. if the backends have been calibrated before, the data shall be plotted as temperatures
4. process all receivers

Note: *RM also suggested to check if the data of several pointing scans could be added together. This feature could be implemented in a later version.*

3.12 * processFocus (*state: open 2001-11-08*)

.. to be done ..

3.13 * processSkidip (*state: open 2001-11-08*)

.. to be done ..

3.14 * processPositionSW (*state: open 2001-11-08*)

.. to be done ..

3.15 * processFrequencySW (*state: open 2001-11-08*)

.. to be done ..

3.16 * processWobblerSW (*state: open 2001-11-08*)

.. to be done ..

3.17 * processOTF (*state: open 2001-11-15*)

Note: *from e-mail HU, 2001-11-15:*

Before a spectral-line on-the-fly (OTF) scan a valid CAL COLD must have been taken.

This CAL COLD scan will be used to calibrate the OTF scan.

During a standard OTF scan:

- 1 the first subscan is taken on an off-source reference position (REF).
- 2 one or several OTF subscans are taken.
- 3 normally another subscan is taken at a reference position.

Optionally: continue with step 2.

Normally the last subscan is a REF;
but in general the sequence can also end with an OTF subscan.

The processing of spectral-line on-the-fly observations shall:

after the first subscan (REF): do nothing

after an OTF subscan that is not the last subscan: do nothing

after the n-th REF subscan (n > 1):
process all OTF subscans taken between the (n-1)-th and n-th
REF subscan, using both the (n-1)-th and the n-th REF subscan.

at the end of the OTF scan, if the last subscan was not a REF:
process all OTF subscans taken since the last REF subscan,
using (only) the last REF subscan.

after any group of OTF subscans has been processed:
generate plots (TBD)

Notes:

Only "standard" OTF scans in the sense explained above shall be
processed correctly by ODP.

ODP requires that REF and OTF data are taken in the same scan.

Desiderata for future iterations:

Support frequency-switched OTF.

Could consider using CAL COLD scans taken after the OTF scan and
processing of non-standard OTF scans.

3.18 * abbaToFits (*state: done 2001-11-15*)

Data of bolometer observations with the Abba backend shall be transformed automatically to the FITS format.

3.19 * heraToFits (*state: done 2001-11-15*)

Data of Hera observations shall be transformed automatically to the FITS format.

3.20 * processWhatElse (*state: open 2001-11-08*)

.. to be done ..

4 Specifications

4.1 * obsProcedure (*state: done 2001-11-07*)

The remote data processing software has to know the observing procedure of a scan. This information has to be recorded in such a way that data processing can also be done later (offline).

4.2 * endOfScan (*state: fixed 2001-11-08*)

The data processing software has to know if a scan has finished or not in order to do "quick look" processing or endOfScan processing. E.g. for pointing "quick look" just is means plotting the recorded data whereas at the end of a pointing scan the pointing corrections are calculated.

4.3 * workingDirectory (*state: fixed 2001-11-09*)

A scheme shall be setup to define a working directory for each post processing task (e.g., oflcal, abbaToFits).

4.4 * scriptCoding (*state: fixed 2001-11-09*)

Coding rules for the post processing scripts shall be defined. I assume that these scripts will be written in SIC.

4.5 * taskControl (*state: fixed 2001-11-09*)

The operator shall be able to restart the postprocessing tasks.

4.6 * taskCrashes (*state: fixed 2001-11-09*)

If a postprocessing task (e.g. OTFCAL) crashes, the operator and observer shall be informed and the task shall be restarted automatically.

5 Design

5.1 * architectureOverview (*state: done 2001-11-13*)

We have the following tasks:

1. data-producer: the current telescope control software produces so-called raw data and header files: header files with specification of the observation and one data file for each backend (see also subsections dataTransfer...). All files are finally stored on mrt-lx1 .
2. post-processing tasks (data-consumer): After raw header and data files are produced, other task will process these files. Possible tasks are:
 - (a) calibration of heterodyne data (opCalibration)
 - (b) fits converters (opAbbaToFits, opHeraToFits)
 - (c) data analysis of pointing, focus, skidip (opPointing, ...)
 - (d) data plots (opPlot)

The "communication" between producer and consumer is based on the <http://www.iram.es/IRAMES/documents/projectPipeline/> . Consumer task inform the producer that they want to be informed about new data. It is up to the consumer to decide what to do with the data. Detail are given below in subsection "dataProcessingPipeline".

5.2 * obsProcedure (*state: done 2001-11-07*)

The OBS program shall write the observing procedure of a scan into the scan header using a new OBSINP command OBSP. The procedure can be up to 8 characters long.

Note: *I has to be checked if more information about the observing procedure has to be provided by OBS.*

5.3 * dataTransferIramEA (*state: done 2001-11-07*)

In the current system, for most backends complete raw data files are generated on iramea . For these backends, all raw data generated is transferred automatically to mrt-ux1 . See also <http://www.iram.es/IRAMES/documents/prRawDataToUnix/> .

5.4 * dataTransferOthers (*state: done 2001-11-07*)

1. 4MHz backend:

- (a) the backend data is transferred directly from the backend processor vbe4m to the file server mrt-ux1 .
- (b) The 4MHz backend processor also send UTC for all data dumps to the backend process beorga on iramea .
- (c) The antenna control software reserves space in the raw data file and adds DAPs for the specific UTC time given.
- (d) When this raw data file is transferred at the end of a subscan to the MRT file server, the actual data is written into the raw data file

2. ABBA:

- (a) Raw data files are only written for ONE channel in wobbler and skydip mode.
- (b) In the fast-scanning mode, data from the continuum backend is recorded every 250ms in order to have DAPs.
- (c) At the end of a subscan:
 - i. the raw data files are transferred
 - ii. data for all channels is retrieved from ABBA
 - iii. FITS files are generated from rowdata and ABBA files

5.5 * endOfScan (*state: open 2001-11-08*)

The antenna control software does not record the total number of subscans in the header. The total number is defined by using 5 parameters (SRPs) plus an option for each of these parameters. We are trying to calculate the total number of subscan from these parameters. An alternative could be to record the total number of subscans by as (as part of the OBSP parameter).

5.6 * processCancelledScans (*state: open 2001-11-08*)

(According to JB Schraml) the preheader of the raw data files have an indication if the subscan was cancelled. The header files are normally written before the end of a subscan (when the first backend is ready to be written) and therefore cannot contain this type of information.

Note: *As a general rule for the NCS it should be foreseen that all subsystems can write information before, during, and after a subscan (or whatever unit is chosen).*

5.7 * processingPipeline (*state: open 2001-11-13*)

1. The data-producer (see also subsection on "architectureOverview") will put raw data of a new subscan into a directory /mrtData/'project'/r'date' .
2. After this, it will put a link to the header file of the new subscan in all subdirectories that of /mrtData/Pipe/raw .

3. Each data processing task has its own subdirectory (e.g. /mrtData/Pipe/raw/cal) and monitors this directory for new links.
4. If a task detects a new link, it does what it has to do and then removes the link.

Please note:

1. data processing tasks can also "forward" links to other tasks, e.g. after a calibration the link can be forwarded to a plot task.
2. the raw data producer can code (e.g. in the file name) what class of observation we have: heterodyne, hera, abba, This can help the consumer to decide if the link has to be processed or can be ignored.
3. The current pipeline concept only foresees consumer that monitor themselves their directory. In the future, we also foresee to have consumers that can be woken up or that are started each time a job is available.

A tool shall display the state of the pipeline: which jobs are pending for how many seconds. The 1st version of the pipeline will have this structure:

```
raw
  calibration
    obsProcedure
      plot
  abbaToFits
    plot (possibly links to raw/cal../obs../plot)
  heraToFits
    plot (possibly links to raw/cal../obs../plot)
```

This means: raw is the generator of tasks in directories calibration, abbaToFits, and heraToFits. Files and links in these subdirectories are processed by processes odpCalibration, odpAbbaToFits, odpHeraToFits. The odp.. processes can produce new tasks in their subdirectories (or other directories).

5.8 * calibration (*state: open 2001-11-13*)

All data of heterodyne receivers shall be processed by the calibration task. The calibration tasks will prepare an OFTCAL script:

1. define SIC variables
 - (a) the observing procedure (odpProcedure)
 - (b) the project(odpProject)
 - (c) scan, subscan number(odpScan, odpSubscan)
 - (d) end of scan,0: no, 1:yes (odpEndOfScan)
 - (e) sky (0) or reference (1) (odpSkyReference)
 - (f) calibration scan (1) or no (0) (odpCalibration)

- (g) scan number of last calibration or -1 (odpCalibrationLast)
- (h) backend number (1,2,3,4,or 7) (odpBackend)
- (i) the directory where the raw data is (odpDirectory)
- (j) the filename of the backend file (odpFileName)

2. call a script with the name of the observing procedure
3. wait for the end of the script WITH a TIMEOUT

Note: *please check if more parameters are needed*

5.9 * feedbackObs (*state: open 2001-11-07*)

Note: *We do not plan to install this feature in the first version.*

OBS will have a global section and a second process (obsServer) shall map to this memory area and run a server software that allows to set/get values of this memory area. OBS uses values from this array to forward the results of the data processing to the antenna control program (internally via OBSINP commands).

5.10 * monitoring (*state: open 2001-11-07*)

The data processing software writes a log record that is monitored on the observer and operator screen (as part as the general monitoring of the MRT). It displays the scan/subscan number, the procedure and possible results.

5.11 * processCalibration (*state: open 2001-11-08*)

Note: *AS will design a script for calibration.*

5.12 * processOTF (*state: open 2001-11-15*)

Note: *from e-mail HU, 2001-11-15:*

after the n-th REF subscan (n>1):

otfcal is called with:

- 1 a pre-script that assigns the appropriate values to the following required SIC GLOBAL variables (all integers)

odp#backendCode	! backend 2 3 4 (5) or 7
odp#calScan	! CAL scan number
odp#flyScan	! OTF scan number
odp#refSubscan1	! REF subscan before OTF subs cans
odp#flySubscanList[1]	! OTF subscan
...	! more OTF subs cans (otional)

```

odp#refSubscan2      ! REF subscan after OTF
2 a script that processes the OTF data as explained in the
  requirements/specifications and according to the values
  of the variables listed above.
3 a "post"-script. unused.

```

at the end of the OTF scan, if the last subscan was not a REF:

```

otfcal is called as above, except that:
1      odp#refSubscan2 = 0      ! must be set to 0 (zero)

```

generate plots (TBD)

5.13 * abbaToFits (*state: done 2001-11-15*)

A script abbaToFits will monitor directory /mrtData/Pipe/raw/abbaToFits for new links to raw data headers. If the observation used the bolometer with Abba it:

1. will transfer the subscan data from the Abba backend
2. and execute the abbaToFits program to generate a Fits file in the directory /mrt-Data/"project"/fits .

Note: *a future version of the script will also allow to do the transformation offline.*

5.14 * heraToFits (*state: done 2001-11-15*)

A script heraToFits will monitor directory /mrtData/Pipe/raw/heraToFits for new links to raw data headers. If the observation used the Hera receiver it: execute the abbaToFits program to generate a Fits file in the directory /mrtData/"project"/fits .

Note: *a future version of the script will also allow to do the transformation offline.*

6 Implementation

6.1 * obsProcedure (*state: done 2001-11-07*)

The OBS software code has been modified and now writes the observing procedure into the header using Obsinp comand OBSP.

6.2 * dataTransferOthers (*state: done 2001-11-07*)

1. 4 MHz backend: the design is implemented. Currently, the 4MHz processor writes data to files on the file-server via NFS. This data is merged with the pseudo raw data files after the transfer to the file server as part of the data transfer software.(source: /mrt-ux1/usr/local/bin/b4merge.icn)
2. Abba: the process abbaToFits.py transfers ABBA data files via FTP and then executes a merger program abbaToFits. (sources in /mrtData/src): TO BE CHECKED
3. Hera: TO BE CHECKED

6.3 * processCalibration (*state: open 2001-11-07*)

the process odpCalibration (source /mrtData/src/CalProc/pipeCal.icn) does:

1. it checks if new links are in /mrtData/Pipe/raw/cal
2. for a new link it checks if the project has changed: if yes the current OTFCAL process will execute script STOP to finish and a new OTFCAL is started
3. a script with the name of the observing procedure is executed
4. the link it removed

The OTFCAL scripts are under development (by AS, HU).

Note: *We plan to replace the existing program by a python script.*

6.4 * dataProcessingAbbaToFits (*state: open 2001-11-07*)

To be done. See also <http://www.iram.es/IRAMES/documents/projectAbbaControl> .

6.5 * dataProcessingHeraToFits (*state: open 2001-11-07*)

To be done. See also <http://www.iram.es/IRAMES/documents/projectHeraControl> .

6.6 * processOTF (*state: open 2001-11-15*)

Note: *from e-mail HU, 2001-11-15:*

```
"odp#prescriptOTF.otf"
```

```
-----
```

Template script showing how to assign values to the required variables. In this template the values are taken from the SIC parameters to 	 in a real case they should be set directly.

Up to 99 OTF subscans can be specified:

```
odp#flySubscanList[1] = i
```

```
...  
odp#flySubscanList[99] = j
```

```
"odp#processOTF.otf"  
-----
```

Process the OTF subscans as explained above:

For each OTF subscan in odp#flySubscanList (for each element of odp#flySubscanList that is > 0) the scripts odp#prescript1OTF.otf and odp#process1OTF.otf are called.

Scripts, Used Scripts, and Helper Scripts:

odp#defineOTF.otf

ensures the correct definition of the SIC GLOBAL variables

odp#examineOTF.otf

makes a neat list of the SIC GLOBAL variables and their values

odp#plot1OTF.class
(pre-prototype, unused)

makes a plot

odp#prescript1OTF.otf

sets values for processing of 1 OTF subscan

odp#prescriptOTF.otf

sets values for processing of OTF subscans from one CRO cycle

odp#process1OTF.otf

processes 1 OTF subscan
the maximum number of OTF "dumps" is set to 1999 in this script;
maybe this should be determined by another variable ...

NOTE: THIS IS THE (ONLY) SCRIPT THAT DOES "REAL WORK".
SHOULD BE CHECKED AND TESTED ON MORE DATA.

odp#processOTF.otf

organizes processing of OTF subscans from one CRO cycle

Implementation Notes:

These scripts assume that a SIC logical "ODP:" is defined pointing to the directory containing the scripts.

All names of these scripts, as well as the GLOBAL SIC variables they use, start with the string "odp#".

Some overhead could be avoided by processing the CAL in odp#processOTF.otf. In the current version the CAL is processed in odp#process1OTF.otf for each OTF subscan (and each backend). For development and test purposes this has the advantage that all the "real work" is encapsulated in one script.

Note:

The prototypes of these scripts are on mrt-ux1 in:
/users/astro/ungerech/ncs30m/onlineDataProcessing

6.7 * abbaToFits (*state: done 2001-11-15*)

TBD

6.8 * heraToFits (*state: done 2001-11-15*)

TBD

7 Installation

7.1 * obsProcedure (*state: done 2001-11-07*)

A new OBS version has been installed as the default version on iramea

that records the observng procedure in the raw data header.

7.2 * processCalibration (*state: open 2001-11-07*)

The process pipeCal has to be started during boot of the file server

8 Logbook

8.1 * meet2001-11-12 (*state: done 2001-11-13*)

Meeting by AS, HU, WB: discussed concepts as written in "predraft" (2001-11-09). HU suggested to:

1. separate calibration and further processing (data plot, pointing, ...)
2. start and stop OTFCAL for each subscan
3. check if calibration of data could also be done with calibrations done after an observation

AS mentions that data of focus, skidip, and calibration are not recorded in CLASS files. We agreed on which steps to do next (see also section "plan").

A List of Requirements/Specifications and Descendants

1.
 - req dataOnlineProcessing fixed 2001-11-08
2.
 - req dataTransferIramEA done 2001-11-07
 - des dataTransferIramEA done 2001-11-07
3.
 - req dataTransferOthers done 2001-11-07
 - des dataTransferOthers done 2001-11-07
 - impl dataTransferOthers done 2001-11-07
4.
 - req processCalibration fixed 2001-11-08
 - des processCalibration open 2001-11-08
 - impl processCalibration open 2001-11-07
 - inst processCalibration open 2001-11-07
5.
 - req processSubscans fixed 2001-11-08
6.
 - req processScan fixed 2001-11-08
7.
 - req processCancelledScans fixed 2001-11-08
 - des processCancelledScans open 2001-11-08
8.
 - req feedbackObs fixed 2001-11-07
 - des feedbackObs open 2001-11-07
9.
 - req logging fixed 2001-11-07
10.
 - req plots fixed 2001-11-07
11.
 - req processPointing open 2001-11-08
12.
 - req processFocus open 2001-11-08
13.
 - req processSkidip open 2001-11-08
14.
 - req processPositionSW open 2001-11-08
15.
 - req processFrequencySW open 2001-11-08
16.
 - req processWobblerSW open 2001-11-08
17.
 - req processOTF open 2001-11-15
 - des processOTF open 2001-11-15
 - impl processOTF open 2001-11-15
18.
 - req abbaToFits done 2001-11-15
 - des abbaToFits done 2001-11-15
 - impl abbaToFits done 2001-11-15
19.
 - req heraToFits done 2001-11-15

- des heraToFits done 2001-11-15
- impl heraToFits done 2001-11-15
- 20. • req processWhatElse open 2001-11-08
- 21. • spec obsProcedure done 2001-11-07
 - des obsProcedure done 2001-11-07
 - impl obsProcedure done 2001-11-07
 - inst obsProcedure done 2001-11-07
- 22. • spec endOfScan fixed 2001-11-08
 - des endOfScan open 2001-11-08
- 23. • spec workingDirectory fixed 2001-11-09
- 24. • spec scriptCoding fixed 2001-11-09
- 25. • spec taskControl fixed 2001-11-09
- 26. • spec taskCrashes fixed 2001-11-09

B References

1. pipeline concept
<http://www.iram.es/IRAMES/documents/projectPipeline/>
2. project rawDataToUnix documentation
<http://www.iram.es/IRAMES/documents/prRawDataToUnix/>
3. Project Abba Control
<http://www.iram.es/IRAMES/documents/projectAbbaControl>
4. Project Hera Control
<http://www.iram.es/IRAMES/documents/projectHeraControl>