SOLAR ORBITER RPW ROC

RPW Operation Centre

BIAS Calibration Software (BICAS) User Manual

ROC-PRO-SFT-SUM-xxxxx-LES
Iss.01, Rev.00

Prepared by:	**Function:**	**Signature:**	**Date**
Erik P G Johansson	BIAS Data Management		02/03/2017
Verified by:	**Function:**	**Signature:**	**Date**
			Dd/mm/yyyy
Approved by:	**Function:**	**Signature:**	**Date**
			Dd/mm/yyyy
For application:	**Function:**	**Signature:**	**Date**
			Dd/mm/yyyy

CLASSIFICATION **PUBLIC** ☒ **RESTRICTED** □

l'Observatoire de Paris LESIA
Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

CNRS-Observatoire de PARIS
Section de MEUDON – LESIA
5, place Jules Janssen
92195 Meudon Cedex – France

Change Record

Issue	Rev.	Date	Authors	Modifications
1	0	26/04/2017	E. Johansson	First draft

Acronym List

Acronym	Definition
BICAS	BIAS Calibration Software
CLI	Command-line Interface
ICD	Interface Control Document
IRF-U	Swedish Institute of Space Physics, Uppsala section
PDID	Process Data ID
PDV	Process Data Variable
RCS	RPW Calibration Software
ROC	RPW Operation Centre
RPW	Radio and Plasma Waves instrument
SUM	Software User Manual
TBW	To be written

Table of Contents

List of Figures

No table of figures entries found.

List of Tables

1 GENERAL

1.1 Scope of the Document

This user manual primarily seeks to describe information practically useful for the installation and running of BICAS s/w including basic errors, and secondarily offer some insights into its inner workings and source code structure to make debugging and modifications easier, including with regards to the calibration. Much of the basic user interface and functioning is specified by the RCS ICD (AD1) and this document will therefore mostly concern itself with functionality *not* specified in the RCS ICD.

1.2 Applicable Documents

This document responds to the requirements of the documents listed in the following table:

Mark	Reference/Iss/Rev	Title of the document	Authors	Date
AD1	ROC-PRO-PIP-ICD-00037/1/0	RPW Calibration Software Interface Control Document	M.Duarte, X.Bonnin	16/11/2016
AD2	ROC-PRO-DAT-NTT-00006-LES/1/0	RPW Data Products	X.Bonnin	23/12/2016
AD3	ROC-TST-GSE-NTT-00017-LES	Data format and metadata definition for the ROC-SGSE data	X. Bonnin	14/10/2016

1.3 Reference Documents

This document is based on the documents listed in the following table:

Mark	Reference/Iss/Rev	Title of the document	Authors	Date
RD1				
RD2				
RD3				
RD4				
RD5				

2 CONVENTIONS

This document uses a special font when referring to computer variables (e.g. CDF global attributes), filenames, directory paths, commands and CLI arguments, e.g. `bicas --help`.

3 DESCRIPTION OF THE SOFTWARE

3.1 Purpose of the software

BICAS is the BIAS team's contribution to the calibration of RPW instrument data. In its *present form* it aims to take electric data at the physical LFR/BIAS and TDS/BIAS interfaces (voltages in physical units) and produce the corresponding calibrated data corresponding to electrical values at the probes, and differential electric values between the probes. In its *future form* it will take *uncalibrated* LFR and TDS data in engineering units and produce the same output as above. All input and output data is provided in the form of a subset of the datasets defined in AD2.

Important: the software is not complete. For example it cannot handle TDS input, and the calibration is still simplistic. See Section 6.1.1 for details. The source code does however contain pieces of future implementations with functionality that is not yet visible from the outside.

3.2 Operations environment

The SUM shall specify in which environment the software is expected to run. Especially, the interfaces with other software and/or systems shall be mentioned.

TBW

3.3 Software design overview

3.3.1 Introduction

BICAS is fundamentally a MATLAB application that is launched from a Unix/Linux bash script.

Practically, BICAS is developed as a part of a larger git repository of software, named "irfu-matlab", focused on MATLAB code and primarily used internally at the IRF-U. The core of BICAS uses the remainder of irfu-matlab as an internal, but separate library of generic functions. Every new official version of BICAS is released as a commit on the "master" branch of irfu-matlab's git repository, but not every new commit on the master branch corresponds to a new BICAS version.

The reading and writing of CDF files is done via the "NASA CDF patch for MATLAB", freely available from NASA GSFC and already included in irfu-matlab and hence in BICAS.

3.3.2 Structure of the source code and its execution

The (MATLAB) source code consists of a few main files which use the remainder of the code as library functions. MATLAB code files are recognized via their filename suffix "`.m`".

`bicas` is a bash script which launches MATLAB and instructs it to execute the MATLAB function `bicas.m`. This wrapper script only handles a few basic tasks: (1) launching the correct installation of MATLAB, (2) extracting and saving log messages from MATLAB's stdout and stderr to file, and (3) handling errors associated with these tasks. It passes all CLI arguments on to `bicas.m` and receives an exit code from it. It only interprets the CLI argument `--config` that specifies the configuration file in order to determine which MATLAB installation to use, and `--log` which specifies the log directory (see AD1).

`bicas.m` does basic initialization and checks, interprets CLI arguments and their syntax, and converts them into data structures that are more easy for code to work with and act on. For processing datasets, the function calls `execute_sw_mode.m` to do the actual processing associated with a s/w mode.

`execute_sw_mode.m` reads the required inputs datasets (CDF files) into analogous MATLAB data structures in RAM and does some checks on the contents. It uses `data_manager.m` to specify the available input data (analogous to input datasets/CDF files) and requests the desired output (analogous to the output datasets/CDF files). If successful, `execute_sw_mode.m` uses the result, to write the corresponding output datasets to disk as CDF files, complemented with some global attributes and while using the master CDF files.

`data_manager.m`, defines a class intended to be instantiated exactly once. This instance is hereafter referred to as the "data manager". The data manager coordinates the actual processing of dataset data in RAM. The data manager maintains an internal (hidden) set of different "process data variables" (PDVs), each one uniquely referenced by a unique string, a "process data ID" (PDID), used internally. The terms PDV and PDID are frequently used in comments and identifiers in the source code. The PDVs are all related to each other in a conceptual "web" (an acyclic directed graph) describing their dependencies on each other. A Special functions, called "processing functions", each describe how to derive a specific PDV from specific sets of other PDVs. That subset of PDVs which (essentially) correspond to input/output datasets have PDIDs which are essentially shortened versions of the official `DATASET_ID` combined with the specific `Skeleton_version` (see AD3).

Initially the data manager's PDVs are all empty. The code that uses the class, the "user", sets some of the PDVs (calls a corresponding method for every such PDV) based on the data the user already has (input datasets). The user of the class then asks the data manager to derive and return specific PDVs (calls a method for every such PDV). For every such request, the data manager tries to derive the requested PDV by recursively deriving unset PDVs from other PDVs internally until it finds PDVs which have already been set by the user.

By using this data manager scheme (`data_manager.m` and `constants.m` described below) the code is able to (1) take advantage of similarities between similar datasets (including multiple skeleton versions of the same dataset), (2) take advantage of similarities in processing between all these datasets, (3) be very flexible, and (4) keep track of a large number of datasets and s/w modes in parallel which is useful during development (inoffical, unfinished modes; ground calibration vs in-flight; quickly disabling/enabling them), (5) make it easier to write automatic test code for the processing (not yet implemented), (6) semi-automatically update the s/w descriptor, (7) semi-automatically update the CLI argument parsing and syntax checking. The scheme is a more purified version of the scheme used for irfu-matlab's processing of MMS data.

dm_processing_functions.m (dm=data manager) is a container for the processing functions used by data_manager.m. It could therefore be said that this is where the actual processing and calibration of bulk data occurs.

constants.m contains metadata associated with each supported input and output dataset, and the s/w modes that tie them together. This includes the data manager PDIDs for specific datasets, information for the s/w descriptor, CLI arguments associated with s/w modes and input data sets, etc.. Disabling/enabling s/w modes here will automatically disable/enable the s/w mode for the entire s/w, including in the s/w descriptor returned by the call

```
    bicas --identification
```

and the s/w modes permitted by the CLI syntax. It is thus a useful way of temporarily hiding and unhiding functionality while still keeping the s/w ICD compliant.

settings.m contains various global configuration settings (incl. constants) which the user might want to be aware of, or modify, to alter the behaviour of BICAS. This includes settings for: the granularity of logging, certain values in the S/W descriptor (e.g. author), configuring extra checks/relaxed checks on input datasets, choosing between implementations, hardcoded values used by temporary solutions etc. Choosing implementations may refer to choosing between old and new algorithms, temporary implementations vs. unfinished/unchecked final implementations (e.g. calibration). Hardcoded values used by temporary(?) solutions may refer to e.g. which data version to use for the output datasets CDF global attribute Data_version; see AD3). The hardcoded default settings can be overridden using CLI arguments (Section 4.2.8). The settings can be viewed by calling bicas --help.

error_safe_constants.m contains a smaller number of global constants that must be available early in the execution and in the event of error, e.g. error codes. The hardcoded required MATLAB version is defined here. There is no provision for modifying these values without altering the source code.

3.4 External interfaces

BICAS contains no user interfaces besides those specified in the RCS ICD (AD1): The command-line arguments (slightly amended; See Section 4.2.8), the config file, the log file, and the standard/error output.

3.5 Software operating modes

Since BICAS is fundamentally implemented in MATLAB, and the bash wrapper script that launches it passes on all the CLI arguments to the corresponding MATLAB script, it can also be run from within the MATLAB command-line with almost identical functionality and user interface. This is useful since (1) it can likely be run on other platforms than Linux(!), (2) it eliminates the overhead of launching and quitting MATLAB (a few seconds), and (3) it enables the user to use MATLAB's debugging features (breakpoints, step-by-step execution, inspecting internal variables etc.). It is therefore very useful for debugging and development. The primary other differences between this mode of operation and launching BICAS from the bash wrapper script are: (1) The user has to use MATLAB's syntax for submitting arguments, and (2) log messages cannot be saved to a log file, and are instead intermingled with error messages (stderr) and standard output (stdout) although they can be visually distinguished using BICAS' own standardized use of prefixes.

To call BICAS from within MATLAB, simply launch `bicas.m` from the MATLAB command-line with the same arguments that would be used from the OS, and using MATLAB's syntax, e.g.

```
bicas('--version')
```

. Flags `--config` and `--log` (both specified in the RCS ICD, AD1) are permitted but will have no effect since only the bash launcher script interprets and uses that information. The value returned by `bicas.m` is identical to the exit code that would be returned to the OS when launching BICAS from the bash launch script.

Running BICAS from the MATLAB command-line may require the MATLAB-internal path variable, used by MATLAB for finding MATLAB code files, to include the path to BICAS' `src/` directory.

3.6 Description of the software data

3.6.1 Software input data

The SUM shall give a list of expected input data when the software runs normally. The list shall at least include a short content description, the purpose and the format of the input data. The mode(s) for which the software needs the inputs shall be also mentioned. The table below can be used as an example. (Note that the RPW Data set ID can be used for the input data name column.)

Input data name/ DATASET_ID	Description	Purpose	Format	Required in the following mode(s)
ROC-SGSE_L2R_RPW-LFR-SBM1-CWF				
ROC-SGSE_L2R_RPW-LFR-SBM2-CWF				
ROC-SGSE_L2R_RPW-LFR-SURV-CWF				
ROC-SGSE_L2R_RPW-LFR-SURV-SWF				
ROC-SGSE_L2R_RPW-TDS-LFM-CWF				
ROC-SGSE_L2R_RPW-TDS-LFM-RSWF				
ROC-SGSE_HK_RPW-BIA				

Table 1. Software input data list.

NOTE:

- *The input data related to the software internal and testing uses shall not be described in this section, excepted if they are required to run the software in very specific cases/modes only.*
- *A reference to the detailed description of the input data shall be supplied. The reference can be a document or a file schema (e.g., CDF skeleton or XML schema). In the case where there is no reference, the full description shall be given in the present document.*

3.6.2 Software output data

The SUM shall give a list of expected output data products when the software runs normally. The list shall at least include a short content description and the format of the data products. The mode(s) for which the software generates the outputs shall be also mentioned. The table below can be used as an example. (Note that the RPW Data set ID can be used for the Output data name column.)

Output data name/ DATASET_ID	Description	Format	Product of the following mode(s)
ROC-SGSE_L2S_RPW-LFR-SBM1-CWF-E			
ROC-			

SGSE_L2S_RPW-LFR-SBM2-CWF-E			
ROC-SGSE_L2S_RPW-LFR-SURV-CWF-E			
ROC-SGSE_L2S_RPW-LFR-SURV-SWF-E			
ROC-SGSE_L2S_RPW-TDS-LFM-CWF-E			
ROC-SGSE_L2S_RPW-TDS-LFM-SWF-E			

Table 2. Software output data list.

NOTE:

- *The output data related to the software internal and testing uses shall not be described in this section.*
- *A reference to the detailed description of the output data shall be supplied. The reference can be a document or a file schema (e.g., CDF skeleton or XML schema). In the case where there is no reference, the full description shall be given in the present document.*

The s/w descriptor returned by invoking

```
bicas --identification
```

is the authoritative list of valid s/w modes and their corresponding input and output datasets. If the above tables contradicts the s/w descriptor referenced above, then the s/w descriptor applies.

TBW

3.6.3 Software installation and configuration files

BICAS uses a configuration file which specifies which MATLAB version to use. This is useful if the user has multiple MATLAB installations on the same machine. The default configuration file is `config/bicas.conf`. The configuration file can be overridden with the `--config` CLI argument.

The OS should preferably set the environment variable CDF_LEAPSECONDSTABLE before invoking BICAS, see Section 4.1.3

3.6.4 Software internal data files

BICAS requires dataset master CDF files under the `data/` subdirectory. Only master CDF files corresponding to datasets that shall be produced are necessary.

3.6.5 Software testing data files

BICAS contains some MATLAB code corresponding to somewhat informal automated test code that tests some well-defined functions, but which only cove small parts of BICAS. The code is found in the files `+tools/*_TEST.m`. They have to be invoked manually from within MATLAB.

4 Reference manual

4.1 Software configuration requirements

4.1.1 General

BICAS requires Linux and MATLAB R2016a to run.

4.1.2 Hardware configuration requirements

BICAS itself does not presently have any particular hardware requirements. Instead, the primary hardware requirements should be driven by MATLAB's requirement and the sizes of the datasets. BICAS (really, irfu-matlab; including BICAS' data files and excluding the git repository data) requires 59 MiB of storage space (03/03/2017). BICAS has been developed and tested on a machine with 16 GiB of RAM but it likely works with less.

4.1.3 Software configuration requirements

Running BICAS the nominal way as specified in the ICD (AD1) requires Linux, the bash shell, and MathWorks MATLAB, version R2016a. BICAS has been developed and tested under openSuse Linux 13.1, 64-bit, and GNU Bash 4.2. The exact Linux version should be unimportant.

BICAS can *likely* be run using MATLAB versions later than R2016a too since new versions of MATLAB are typically backwards compatible, and *possibly* also using slightly older MATLAB versions, but neither of these two options are not guaranteed to work. In both of these cases, one has to manually alter BICAS' hard-coded MATLAB version check. See also Section 3.4 for running BICAS using only MATLAB (on any OS).

BICAS needs a leap second table to function through its use of the "NASA CDF patch for MATLAB". If (1) there is an OS environment variable `CDF_LEAPSECONDSTABLE` that specifies a valid path to a valid leap second table (preferred), then that will be used. If it does not, BICAS will use (2) the file `CDFLeapSeconds.txt` which is included in BICAS (via irfu-matlab). If both are available and the latter (2) seems more up-to-date than the former (1), then (1), `CDF_LEAPSECONDSTABLE,` will still be used but a warning will be printed in the log.

Although BICAS may or may not be delivered with git repository data (a `.git/` directory), it does *not* require the git repository data, and it does *not* require git to be installed and used.

4.2 Operations manual

4.2.1 Set- up and initialisation

The SUM shall describe any procedures to be performed by the user in order to install software on the equipment, to perform the installation, to configure the software, to delete or overwrite former files or data, and to enter parameters for software operation.

4.2.2 Getting started

The SUM shall include the step- by- step procedures for beginning work, including any options available, and a check- list for problem determination.

4.2.3 Normal operations

The SUM shall identify the normal operations, to be performed by the user, for the use of software, including inputs from other software or hardware that may affect the software's interface with the system, outputs, diagnostic or error messages or alarms.

4.2.4 Normal termination

The SUM shall describe how the user can cease or interrupt use of the software and how to determine whether normal termination or cessation has occurred.

4.2.5 Error conditions

The SUM shall describe the common error conditions that can occur as a result of executing a function, and how to detect that the error has occurred.

4.2.6 Recover runs

No recovery operation should be needed to re-run BICAS after error, besides possibly deleting corrupt output datasets produced by the s/w before the failure (unlikely).

4.2.7 Help method

The SUM shall include the help information about method installation, in terms of the actions to be performed by the user, how to invoke a function, possible errors, how to resolve them and what results to expect.

4.2.8 Commands and operations

BICAS only receives input from the user via its CLI arguments. It can receive no user input after its invocation. The syntax for the CLI arguments, here referred to as just the "syntax", is compatible with the RCS ICD (AD1). The command

```
bicas --help
```

, which is specified in the RCS ICD (AD1), prints the possible error codes, the settings that exist and the values which are actually used by the s/w. This includes settings which are overwritten by optional `–setting` options explained below.

BICAS supports the optional CLI option

```
--setting <key> <value>
```

. `<key>` is a string referencing an internal setting. The setting must be defined in `settings.m`. `<value>` is a string representation of the specified setting's new value. The value specified here overrides the default value in `settings.m`. This flag and its two arguments may appear at an arbitrary position in the sequence of flags except that it must not be the first argument. It may appear an arbitrary number of times (zero to infinity).

4.2.9 Error messages

The SUM shall list all error messages, diagnostic messages, and information messages that can occur while accomplishing any of the user's functions, including the meaning of each message and the action to be taken after each such message.

4.2.10 Software testing operations

The s/w does not have any testing mode.

4.2.11 Software upgrading operations

BICAS does not have any facilities for automatic updates. Updating the software should be identical to removing the old version and installing the new one, with the exception of keeping the configuration file and master CDF files.

4.2.12 Software uninstalling operations

BICAS itself does not require any particular uninstallation procedure. The directory and files can simply be deleted.

Uninstalling MATLAB may in principle require some form of uninstallation procedure. See MATLAB documentation for help. The uninstallation procedure for MATLAB R2016a on Linux should be to simply delete the MATLAB installation directory.

5 Tutorial

5.1 Introduction

The SUM shall describe how to use the software and what the software does, combining tutorials and reference information for both novices and experts.

5.2 Getting started

The SUM shall include a welcoming introduction to the software.

5.3 Using the software on a typical task

The SUM shall describe a typical use case of the software. Graphical pictures and diagrams to demonstrate the actions performed by the user can be used.

6 Appendices

6.1 Troubleshooting & know issues

6.1.1 Known limitations, caveats

This section lists known current limitations and caveats. Most of the items are likely less important. Most or all of these limitations represent features that should be implemented eventually. Within each category, entries are sorted from most important to least important.

List last updated 17/11/2017.

- Reading CDF files:
    - BICAS might not check explicitly if an input CDF file (depends on setting), or master CDF file appears to be of the right type of dataset (dataset ID) before trying to process the information. Note: Many input datasets (i.e. *not* generated by BICAS) found in the wild have not set the CDF global attributes needed to pass this check.
    - Can neither read, process, nor write TDS datasets.
- Processing:
    - Voltage calibration only uses proportionality factors, i.e.:
        - No calibration for different frequency response are used.
        - No voltage offsets are compensated for.
    - Bias currents are not known. Therefore, zVariables `IBIAS1-IBIAS3` are set to fill values.
    - LFR: Algorithms can use `ACQUISITION_TIME` for time rather than Epoch (for comparing HK data time stamps to SCI timestamps). Depends on setting.
    - LFR: zVariables `QUALITY_BITMASK`, `QUALITY_FLAG` are set to fill values with the right size when the source dataset has empty zVariables (instead of copying).
    - Mux/demux modes 5-7 (calibration modes 1-2) have not been implemented.
    - The conversions between tt2000 (Epoch), `ACQUISITION_TIME` and UTC have not been verified.
    - Many combinations of input dataset CDF zVariables (settings) are untested even if they should work in theory.
- Writing CDF files:
    - Output CDF files only use fill values correctly for floating-point numeric zVariables.
    - Global attributes:
        - `Generation_date`: Date is set in local time rather than UTC.
        - Some attributes which should be set, and which are not inherited from the master CDF files, are not set, e.g. `SPECTRAL_RANGE_MIN`, `SPECTRAL_RANGE_MAX`, `TIME_MIN`, `TIME_MAX`.
        - Note: Failure to set global attributes can also be due to missing global attribute values in the parent datasets.

    - Some global attributes may in principle be wrong if using a s/w mode with multiple output datasets (hypothetical since no such s/w mode exists yet).
    - (Sets some global attributes to the same value for all datasets).
  - CDF Epoch's zVariable attributes `SCALEMIN`, `SCALEMAX` are not set correctly.
  - BICAS cannot produce CDF files with zVariables with zero records.

7 LIST OF TBC/TBD/TBWS

TBC/TBD/TBW			
Reference/Page/Location	Description	Type	Status

8 DISTRIBUTION LIST

LISTS See Contents lists in “Baghera Web”: Project’s informations / Project’s actors / RPW_actors.xls and tab with the name of the list or NAMES below		Tech_LESIA
		Tech_MEB
		Tech_RPW
		[Lead-]Cols
		Science-Cols

INTERNAL

LESIA CNRS		

LESIA CNRS		

EXTERNAL (To modify if necessary)

CNES		C. FIACHETTI
		C. LAFFAYE
		R.LLORCA-CEJUDO
		E.LOURME
		M-O. MARCHE
		E.GUILHEM
		J.PANH
		B.PONTET
IRFU		L. BYLANDER
		C.CULLY
		A.ERIKSSON
		SE.JANSSON
		A.VAIVADS
LPC2E		P. FERGEAU
		G. JANNET
		T.DUDOK de WIT
		M. KRETZSCHMAR
		V. KRASNOSSELSKIKH
SSL		S.BALE

AsI/CSRC		J.BRINEK
		P.HELLINGER
		D.HERCIK
		P.TRAVNICEK
IAP		J.BASE
		J. CHUM
		I. KOLMASOVA
		O.SANTOLIK
		J. SOUCEK
		L.UHLIR
IWF		G.LAKY
		T.OSWALD
		H. OTTACHER
		H. RUCKER
		M.SAMPL
		M. STELLER
LPP		T.CHUST
		A. JEANDET
		P.LEROY
		M.MORLOT