The Data Access Library (DAL)

Lars Bähren

Radboud Universiteit Nijmegen
Afdeling Sterrenkunde

LOFAR/HDF5 Meeting
09 Sep 2010

Motivations for creating the DAL

- Abstraction of details specific to a set of data formats
  - FITS
  - MeasurementSet

Motivations for creating the DAL

- Abstraction of details specific to a set of data formats
  - FITS
  - MeasurementSet
- Common interface to access conceptually similar data structures
  - Group
  - Attribute / Keyword
  - Table
  - Array of datapoints

Motivations for creating the DAL

- Abstraction of details specific to a set of data formats
  - FITS
  - MeasurementSet
- Common interface to access conceptually similar data structures
  - Group
  - Attribute / Keyword
  - Table
  - Array of datapoints
- Reference implementation for creation of and access to LOFAR standard data products
  - new types of data products
  - support for complex data model matching experiment characteristics

Requirements

- C++ library with a small amount of external package dependencies
  - DAL embedded in LUS, to handle external dependencies
  - object-oriented design

Requirements

- C++ library with a small amount of external package dependencies
  - DAL embedded in LUS, to handle external dependencies
  - object-oriented design
- Python bindings wrapping the functionality of the C++ library

Requirements

- C++ library with a small amount of external package dependencies
  - DAL embedded in LUS, to handle external dependencies
  - object-oriented design
- Python bindings wrapping the functionality of the C++ library
- portable across various platforms
  - build from source
  - cross-platform build system → *CMake* Cross-Platform Makefile Generator

DAL library components

- `core`
  - common methods
  - definition of types
  - data format abstraction

DAL library components

- core
  - common methods
  - definition of types
  - data format abstraction
- coordinates
  - representation of world coordinates

DAL library components

- `core`
  - common methods
  - definition of types
  - data format abstraction
- `coordinates`
  - representation of world coordinates
- `data_common`
  - definition of interfaces for building blocks from which to construct LOFAR standard data products

DAL library components

- core
  - common methods
  - definition of types
  - data format abstraction
- coordinates
  - representation of world coordinates
- data_common
  - definition of interfaces for building blocks from which to construct LOFAR standard data products
- data_hl
  - high-level interfaces to LOFAR standard data products

DAL library components

- `core`
  - common methods
  - definition of types
  - data format abstraction
- `coordinates`
  - representation of world coordinates
- `data_common`
  - definition of interfaces for building blocks from which to construct LOFAR standard data products
- `data_hl`
  - high-level interfaces to LOFAR standard data products
- `bindings`
  - Bindings to the Python scripting language

DAL external dependencies

- Data format libraries
  - HDF5 library
  - CFITSIO
  - CASACORE

DAL external dependencies

- Data format libraries
  - HDF5 library
  - CFITSIO
  - CASACORE
- Python bindings (optional)
  - Python
  - Boost
  - num_util

DAL external dependencies

- Data format libraries
  - HDF5 library
  - CFITSIO
  - CASACORE
- Python bindings (optional)
  - Python
  - Boost
  - num_util
- Misc. 3rd party
  - WCSLIB
  - MPI
  - MySQL

DAL external dependencies

- Data format libraries
  - HDF5 library
  - CFITSIO
  - CASACORE
- Python bindings (optional)
  - Python
  - Boost
  - num_util
- Misc. 3rd party
  - WCSLIB
  - MPI
  - MySQL
- Configuration & build
  - CMake

DAL distribution & installation

- Part of LOFAR User Software (LUS)

DAL distribution & installation

- Part of LOFAR User Software (LUS)
- Source available through SVN repository (public readable)

```
svn co http://usg.lofar.org/svn/code/trunk lofarsoft
```

DAL distribution & installation

- Part of LOFAR User Software (LUS)
- Source available through SVN repository (public readable)

```
svn co http://usg.lofar.org/svn/code/trunk lofarsoft
```

- External package dependencies handled through build system

```
cd build
./bootstrap
make dal
```

DAL distribution & installation

- Part of LOFAR User Software (LUS)
- Source available through SVN repository (public readable)

```
svn co http://usg.lofar.org/svn/code/trunk lofarsoft
```

- External package dependencies handled through build system

```
cd build
./bootstrap
make dal
```

- Installed components
  - libdal – C++ library
  - pydal – Python module
  - Header files
  - Executables (application & test programs)

DAL test builds

No file changed as of **Wednesday, September 08 2010 07:00:00 CEST** Help

[Show Filters]

Nightly

Site	Build Name	Update		Configure			Build			Test				Build Time
		Files	Min	Error	Warn	Min	Error	Warn	Min	NotRun	Fail	Pass	Min	
lfe001	Linux-c++	0	0	0	0	0	0	22	9.8	0	7	60	0.9	2010-09-09T05:19:51 CEST
Totals	**1 Builds**	0	0	**0**	**0**	0	**0**	**22**	9.8	**0**	**7**	**60**	0.9	

No Continuous Builds

Experimental

Site	Build Name	Update		Configure			Build			Test				Build Time
		Files	Min	Error	Warn	Min	Error	Warn	Min	NotRun	Fail	Pass	Min	
lce063	Linux-c++	0	0	0	0	0.1	0	0	0.1	1	18	65	1	2010-09-09T05:34:53 CEST
lce063	Linux-c++	0	0	0	0	0.1	0	0	0.1	1	18	65	1	2010-09-09T02:34:07 CEST
lce063	Linux-c++	0	0	0	0	0	0	0 -22	0.1	1	18	65	1	2010-09-08T23:35:00 CEST
dop143	Linux-c++	0	0	0	0	0	36 +36	35 -15	1.6	0	13	54	5.9	2010-09-08T19:26:55 CEST

Standard data products defined through ICDs:

- TBB Time-Series Data
  - raw signal: digitized voltages from individual dipole antennas
  - match hierachical structure of physical elements

Standard data products defined through ICDs:

- TBB Time-Series Data
  - raw signal: digitized voltages from individual dipole antennas
  - match hierachical structure of physical elements
- Beam-Formed Data
  - joint (correlated) signals from multiple dipoles / stations

Standard data products defined through ICDs:

- TBB Time-Series Data
  - raw signal: digitized voltages from individual dipole antennas
  - match hierachical structure of physical elements
- Beam-Formed Data
  - joint (correlated) signals from multiple dipoles / stations
- Radio Sky Image Cubes

Standard data products defined through ICDs:

- TBB Time-Series Data
  - raw signal: digitized voltages from individual dipole antennas
  - match hierachical structure of physical elements
- Beam-Formed Data
  - joint (correlated) signals from multiple dipoles / stations
- Radio Sky Image Cubes
- Dynamic Spectrum Data

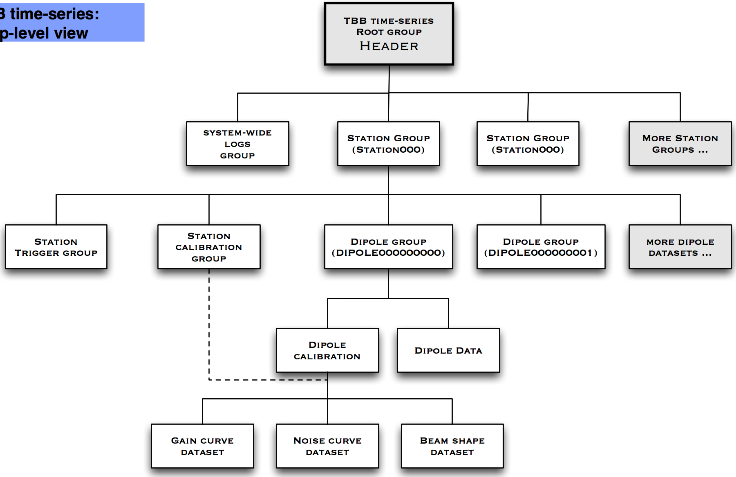
Standard data products defined through ICDs:

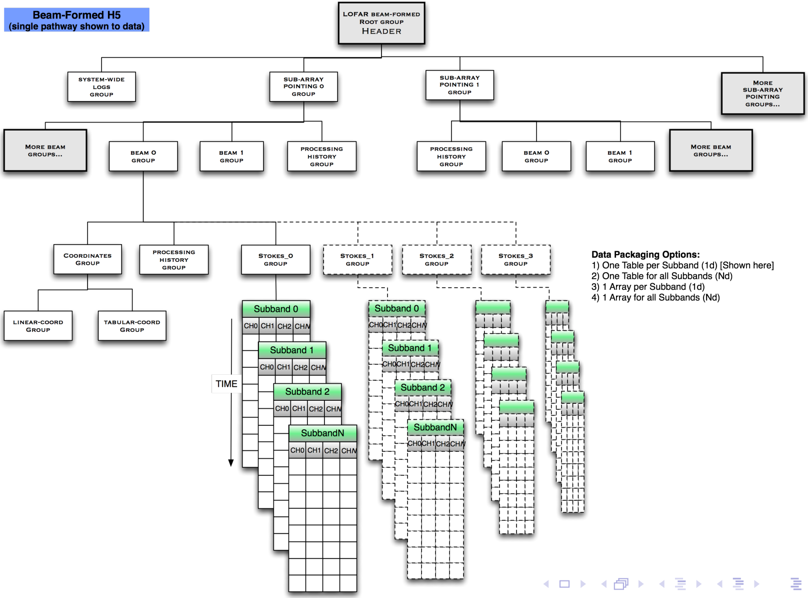
- TBB Time-Series Data
  - raw signal: digitized voltages from individual dipole antennas
  - match hierachical structure of physical elements
- Beam-Formed Data
  - joint (correlated) signals from multiple dipoles / stations
- Radio Sky Image Cubes
- Dynamic Spectrum Data
- Visibility Data

Standard data products defined through ICDs:

- TBB Time-Series Data
  - raw signal: digitized voltages from individual dipole antennas
  - match hierachical structure of physical elements
- Beam-Formed Data
  - joint (correlated) signals from multiple dipoles / stations
- Radio Sky Image Cubes
- Dynamic Spectrum Data
- Visibility Data
- Rotation Measure Synthesis Cubes

TBB time-series: top-level view





Data Packaging Options:
1) One Table per Subband (1d) [Shown here]
2) One Table for all Subbands (Nd)
3) 1 Array per Subband (1d)
4) 1 Array for all Subbands (Nd)

Design approach

- modular, object-oriented implementation

Design approach

- modular, object-oriented implementation
  - one C++ class per type of group / dataset

Design approach

- modular, object-oriented implementation
  - one C++ class per type of group / dataset
  - common basic interface for all object classes

Design approach

- modular, object-oriented implementation
  - one C++ class per type of group / dataset
  - common basic interface for all object classes
  - hide low-level library call behind reasonably simple API

Design approach

- modular, object-oriented implementation
  - one C++ class per type of group / dataset
  - common basic interface for all object classes
  - hide low-level library call behind reasonably simple API
- recursive creation and transversal of hierarchical structures

Design approach

- modular, object-oriented implementation
  - one C++ class per type of group / dataset
  - common basic interface for all object classes
  - hide low-level library call behind reasonably simple API
- recursive creation and transversal of hierarchical structures
- allow working on sub-trees of hierachical structure

DAL dataset support – high-level interfaces

Classes

class	DAL::BeamFormed High-level interface between beam-formed data and the DAL. More...
class	DAL::BeamGroup High-level interface between beam-formed data and the DAL. More...
class	DAL::BeamSubband High-level interface between beam-formed data and the DAL. More...
class	DAL::BF_Beam High-level interface to the station beam of a BF dataset. More...
class	DAL::BF_Dataset High-level interface to the root-group of a beamformed dataset. More...
class	DAL::BF_PrimaryPointing High-level interface to the Primary Pointing Direction of a BF dataset. More...
class	DAL::BF_ProcessingHistory High-level interface to the processing history attached to a BF dataset. More...
class	DAL::BFRaw High-level interface between raw beam-formed data and the DAL. More...
class	DAL::ITS_ExperimentMeta Storage of meta information from an LOFAR ITS experiment. More...
class	DAL::LOPES_EventFile Read in LOPES event files. More...
class	DAL::SysLog High-level interface to the system logs attached to a beamformed dataset. More...

data_common example (1)

```
class BF_Dataset : public CommonInterface {

  //! Name of the data file
  std::string filename_p;
  //! LOFAR common attributes attached to the root group of the dataset
  CommonAttributes commonAttributes_p;
  //! Sub-Array Pointings
  std::map<std::string,BF_SubArrayPointing> subArrayPointings_p;
  //! Container for system-wide logs
  std::map<std::string,SysLog> sysLog_p;

public:

  //! Default constructor
  BF_Dataset (std::string const &filename);

  //! Argumented constructor
  BF_Dataset (DAL::Filename &infile,
              bool const &create=true);

 ...

  //! Open a beam group
  bool openBeam (unsigned int const &pointingID,
                 unsigned int const &beamID,
                 bool const &create=true);
```

data_common example (2)

```
Filename file ("123456789", "test", Filename::bf, Filename::h5);

BF_Dataset dataset (file);

/* open sub-array pointing groups, create if they do not exist yet */
dataset.openSubArrayPointing(0,true);
dataset.openSubArrayPointing(1,true);
dataset.openSubArrayPointing(2,true);

/* open beams residing in an existing group */
dataset.openBeam(0,0,true);
dataset.openBeam(0,1,true);

/* open beams residing in a not yet existing group */
dataset.openBeam(10,0,true);
dataset.openBeam(10,1,true);

/* Extract sub-group */
DAL::BF_SubArrayPointing pointing = dataset.subArrayPointing (0);
```

DAL dataset support – common functionality

Classes

class	**DAL::CommonAttributes** Collection of attributes common to all LOFAR datasets. More...
class	**DAL::CommonInterface** Common functionality for the high-level interfaces to the datasets. More...
class	**DAL::Filename** Class to generate filenames matching the LOFAR convention. More...
class	**DAL::HDF5Dataset** A class to encapsulate the operations required to work with a HDF5 dataset. More...
class	**DAL::HDF5Hyperslab** A hyperslab region for selective access to a dataspace. More...
class	**DAL::HDF5Property** Brief description for class **HDF5Property**. More...
class	**DAL::HDF5Table** Brief description for class **HDF5Table**. More...
class	**DAL::SAS_Settings** Brief description for class **SAS_Settings**. More...
class	**DAL::Timestamp** Wrapper for the time information in its various formats. More...
class	**DAL::HDF5Attribute** Collection of methods to deal with HDF5 attributes. More...

data_hl example DAL::CommonInterface

- common functionality for the high-level interfaces to the datasets

data_hl example DAL::CommonInterface

- common functionality for the high-level interfaces to the datasets
- infrastructure for placing/opening a group/dataset within a file

```
bool open (hid_t const &location,
           std::string const &name,
           bool const &create)
```

data_hl example DAL::CommonInterface

- common functionality for the high-level interfaces to the datasets
- infrastructure for placing/opening a group/dataset within a file

```
bool open (hid_t const &location,
           std::string const &name,
           bool const &create)
```

- common method to get/set attributes

```
template <class T> bool getAttribute (std::string const &name,
                                       T &val)

template <class T> bool setAttribute (std::string const &name,
                                       T const &val)
```

data_hl example DAL::HDF5Dataset

- encapsulate the operations required to work with a HDF5 dataset

```
HDF5Dataset (hid_t const &location,
             std::string const &name,
             std::vector<hsize_t> const &shape,
             hid_t const &datatype)
```

data_hl example DAL::HDF5Dataset

- encapsulate the operations required to work with a HDF5 dataset

```
HDF5Dataset (hid_t const &location,
             std::string const &name,
             std::vector<hsize_t> const &shape,
             hid_t const &datatype)
```

- (partial) read of the data

```
template <class T> bool readData (T data[],
                                  HDF5Hyperslab &slab)

template <class T> bool writeData (T const data[],
                                   std::vector<int> const &start,
                                   std::vector<int> const &count,
                                   std::vector<int> const &block)
```

  - automatically create hyperslab
  - dynamically grow data at write if required

data_hl example DAL::HDF5Hyperslab

- hyperslab region for selective access to a dataspace

data_hl example DAL::HDF5Hyperslab

- hyperslab region for selective access to a dataspace
- book-keeping of hyperslab parameters

```
HDF5Hyperslab (std::vector<int> const &start,
               std::vector<int> const &stride,
               std::vector<int> const &count,
               std::vector<int> const &block,
               H5S_seloper_t const &selection)
```

data_hl example DAL::HDF5Hyperslab

- hyperslab region for selective access to a dataspace
- book-keeping of hyperslab parameters

```
HDF5Hyperslab (std::vector<int> const &start,
               std::vector<int> const &stride,
               std::vector<int> const &count,
               std::vector<int> const &block,
               H5S_seloper_t const &selection)
```

- service functions

```
bool setGap (std::vector<int> const &gap)

unsigned int nofDatapoints ()

std::vector<hsize_t> end ()

bool setHyperslab (hid_t &datasetID,
                   hid_t &dataspaceID,
                   bool const &resizeDataset)
```

World Coordinates

- coordinates that serve to locate a measurement in some multidimensional parameter space

World Coordinates

- coordinates that serve to locate a measurement in some multidimensional parameter space
- measurable quantity such as
  - time
  - length / distance
  - frequency or wavelength associated with a point in a spectrum
  - longitude and latitude in a conventional spherical coordinate system which define a direction in space
  - projection of spherical coordinates onto 2-dim space

World Coordinates

- coordinates that serve to locate a measurement in some multidimensional parameter space
- measurable quantity such as
  - time
  - length / distance
  - frequency or wavelength associated with a point in a spectrum
  - longitude and latitude in a conventional spherical coordinate system which define a direction in space
  - projection of spherical coordinates onto 2-dim space
- enumerations
  - "Stokes parameters" to describe the polarization properties of an electromagnetic wave
  - ID of a channel / dipole / ...

Representations in FITS

A&A 395, 1061–1075 (2002)
DOI: 10.1051/0004-6361:20021326

Astronomy & Astrophysics

Representations of world coordinates in FITS

E. W. Greisen[1] and M. R. Calabretta[2]

[1] National Radio Astronomy Observatory, PO Box O, Socorro, NM 87801-0387, USA
[2] Australia Telescope National Facility, PO Box 76, Epping, NSW 1710, Australia

Received 24 July 2002 / Accepted 9 September 2002

Abstract. The initial descriptions of the FITS format provided a simplified method for describing the physical coordinate values of the image pixels, but deliberately did not specify any of the detailed conventions required to convey the complexities of actual image coordinates. Building on conventions in wide use within astronomy, this paper proposes general extensions to the original methods for describing the world coordinates of FITS data. In subsequent papers, we apply these general conventions to the methods by which spherical coordinates may be projected onto a two-dimensional plane and to frequency/wavelength/velocity coordinates.

Key words. methods: data analysis – techniques: image processing – astronomical data bases: miscellaneous

A&A 395, 1077–1122 (2002)
DOI: 10.1051/0004-6361:20021327

Astronomy & Astrophysics

Representations of celestial coordinates in FITS

M. R. Calabretta[1] and E. W. Greisen[2]

[1] Australia Telescope National Facility, PO Box 76, Epping, NSW 1710, Australia
[2] National Radio Astronomy Observatory, PO Box O, Socorro, NM 87801-0387, USA

Received 24 July 2002 / Accepted 9 September 2002

Abstract. In Paper I, Greisen & Calabretta (2002) describe a generalized method for assigning physical coordinates to FITS image pixels. This paper implements this method for all spherical map projections likely to be of interest in astronomy. The new methods encompass existing informal FITS spherical coordinate conventions and translations from them are described. Detailed examples of header interpretation and construction are given.

Key words. methods: data analysis – techniques: image processing – astronomical data bases: miscellaneous – astrometry

A&A 446, 747–771 (2006)
DOI: 10.1051/0004-6361:20053818

Astronomy & Astrophysics

Representations of spectral coordinates in FITS

E. W. Greisen[1], M. R. Calabretta[2], F. G. Valdes[3], and S. L. Allen[4]

[1] National Radio Astronomy Observatory, PO Box O, Socorro, NM 87801-0387, USA
e-mail: egreisen@nrao.edu
[2] Australia Telescope National Facility, PO Box 76, Epping, NSW 1710, Australia
[3] National Optical Astronomy Observatories, PO Box 26732, Tucson, AZ 85719, USA
[4] UCO/Lick Observatory, University of California, Santa Cruz, CA 95064, USA

Received 12 July 2005 / Accepted 5 October 2005

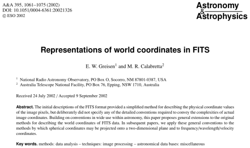
ABSTRACT

Greisen & Calabretta (2002, A&A, 395, 1061) describe a generalized method for specifying the coordinates of FITS data samples. Following that general method, Calabretta & Greisen (2002, A&A, 395, 1077) describe detailed conventions for defining celestial coordinates as they are projected onto a two-dimensional plane. The present paper extends the discussion to the spectral coordinates of wavelength, frequency, and velocity. World coordinate functions are defined for spectral axes sampled linearly in wavelength, frequency, or velocity, linearly in the logarithm of wavelength or frequency, as projected by ideal dispersing elements, and as specified by a lookup table.

Key words. methods: data analysis – techniques: image processing – techniques: radial velocities – techniques: spectroscopic – astronomical data bases: miscellaneous

- Series of seminal papers which are part of the FITS standard

Representations in FITS


A&A 395, 1061–1075 (2002)
DOI: 10.1051/0004-6361:20021326
© ESO 2002

Astronomy & Astrophysics

Representations of world coordinates in FITS

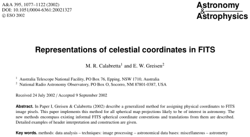
E. W. Greisen[1] and M. R. Calabretta[2]

[1] National Radio Astronomy Observatory, PO Box O, Socorro, NM 87801-0387, USA
[2] Australia Telescope National Facility, PO Box 76, Epping, NSW 1710, Australia

Received 24 July 2002 / Accepted 9 September 2002

Abstract. The initial descriptions of the FITS format provided a simplified method for describing the physical coordinate values of the image pixels, but deliberately did not specify any of the detailed conventions required to convey the complexities of actual image coordinates. Building on conventions in wide use within astronomy, this paper proposes general extensions to the original methods for describing the world coordinates of FITS data. In subsequent papers, we apply these general conventions to the methods by which spherical coordinates may be projected onto a two-dimensional plane and to frequency/wavelength/velocity coordinates.

Key words. methods: data analysis – techniques: image processing – astronomical data bases: miscellaneous


A&A 395, 1077–1122 (2002)
DOI: 10.1051/0004-6361:20021327
© ESO 2002

Astronomy & Astrophysics

Representations of celestial coordinates in FITS

M. R. Calabretta[1] and E. W. Greisen[2]

[1] Australia Telescope National Facility, PO Box 76, Epping, NSW 1710, Australia
[2] National Radio Astronomy Observatory, PO Box O, Socorro, NM 87801-0387, USA

Received 24 July 2002 / Accepted 9 September 2002

Abstract. In Paper I, Greisen & Calabretta (2002) describe a generalized method for assigning physical coordinates to FITS image pixels. This paper implements this method for all spherical map projections likely to be of interest in astronomy. The new methods encompass existing informal FITS spherical coordinate conventions and translations from them are described. Detailed examples of header interpretation and construction are given.

Key words. methods: data analysis – techniques: image processing – astronomical data bases: miscellaneous – astrometry


A&A 446, 747–771 (2006)
DOI: 10.1051/0004-6361:20053818
© ESO 2006

Astronomy & Astrophysics

Representations of spectral coordinates in FITS

E. W. Greisen[1], M. R. Calabretta[2], F. G. Valdes[3], and S. L. Allen[4]

[1] National Radio Astronomy Observatory, PO Box O, Socorro, NM 87801-0387, USA
e-mail: egreisen@nrao.edu
[2] Australia Telescope National Facility, PO Box 76, Epping, NSW 1710, Australia
[3] National Optical Astronomy Observatories, PO Box 26732, Tucson, AZ 85719, USA
[4] UCO/Lick Observatory, University of California, Santa Cruz, CA 95064, USA

Received 12 July 2005 / Accepted 5 October 2005

ABSTRACT

Greisen & Calabretta (2002, A&A, 395, 1061) describe a generalized method for specifying the coordinates of FITS data samples. Following that general method, Calabretta & Greisen (2002, A&A, 395, 1077) describe detailed conventions for defining celestial coordinates as they are projected onto a two-dimensional plane. The present paper extends the discussion to the spectral coordinates of wavelength, frequency, and velocity. World coordinate functions are defined for spectral axes sampled linearly in wavelength, frequency, or velocity, linearly in the logarithm of wavelength or frequency, as projected by ideal dispersing elements, and as specified by a lookup table.

Key words. methods: data analysis – techniques: image processing – techniques: radial velocities – techniques: spectroscopic – astronomical data bases: miscellaneous

- Series of seminal papers which are part of the FITS standard
- conversion of pixel coordinates to world coordinates is regarded as a multi-step process

Representations in FITS

A&A 395, 1061–1075 (2002)
DOI: 10.1051/0004-6361:20021326

Astronomy & Astrophysics

Representations of world coordinates in FITS

E. W. Greisen[1] and M. R. Calabretta[2]

[1] National Radio Astronomy Observatory, PO Box O, Socorro, NM 87801-0387, USA
[2] Australia Telescope National Facility, PO Box 76, Epping, NSW 1710, Australia

Received 24 July 2002 / Accepted 9 September 2002

Abstract. The initial descriptions of the FITS format provided a simplified method for describing the physical coordinate values of the image pixels, but deliberately did not specify any of the detailed conventions required to convey the complexities of actual image coordinates. Building on conventions in wide use within astronomy, this paper proposes general extensions to the original methods for describing the world coordinates of FITS data. In subsequent papers, we apply these general conventions to the methods by which spherical coordinates may be projected onto a two-dimensional plane and to frequency/wavelength/velocity coordinates.

Key words. methods: data analysis – techniques: image processing – astronomical data bases: miscellaneous

A&A 395, 1077–1122 (2002)
DOI: 10.1051/0004-6361:20021327

Astronomy & Astrophysics

Representations of celestial coordinates in FITS

M. R. Calabretta[1] and E. W. Greisen[2]

[1] Australia Telescope National Facility, PO Box 76, Epping, NSW 1710, Australia
[2] National Radio Astronomy Observatory, PO Box O, Socorro, NM 87801-0387, USA

Received 24 July 2002 / Accepted 9 September 2002

Abstract. In Paper I, Greisen & Calabretta (2002) describe a generalized method for assigning physical coordinates to FITS image pixels. This paper implements this method for all spherical map projections likely to be of interest in astronomy. The new methods encompass existing informal FITS spherical coordinate conventions and translations from them are described. Detailed examples of header interpretation and construction are given.

Key words. methods: data analysis – techniques: image processing – astronomical data bases: miscellaneous – astrometry

A&A 446, 747–771 (2006)
DOI: 10.1051/0004-6361:20053818

Astronomy & Astrophysics

Representations of spectral coordinates in FITS

E. W. Greisen[1], M. R. Calabretta[2], F. G. Valdes[3], and S. L. Allen[4]

[1] National Radio Astronomy Observatory, PO Box O, Socorro, NM 87801-0387, USA
e-mail: egreisen@nrao.edu
[2] Australia Telescope National Facility, PO Box 76, Epping, NSW 1710, Australia
[3] National Optical Astronomy Observatories, PO Box 26732, Tucson, AZ 85719, USA
[4] UCO/Lick Observatory, University of California, Santa Cruz, CA 95064, USA

Received 12 July 2005 / Accepted 5 October 2005

ABSTRACT

Greisen & Calabretta (2002, A&A, 395, 1061) describe a generalized method for specifying the coordinates of FITS data samples. Following that general method, Calabretta & Greisen (2002, A&A, 395, 1077) describe detailed conventions for defining celestial coordinates as they are projected onto a two-dimensional plane. The present paper extends the discussion to the spectral coordinates of wavelength, frequency, and velocity. World coordinate functions are defined for spectral axes sampled linearly in wavelength, frequency, or velocity, linearly in the logarithm of wavelength or frequency, as projected by ideal dispersing elements, and as specified by a lookup table.

Key words. methods: data analysis – techniques: image processing – techniques: radial velocities – techniques: spectroscopic – astronomical data bases: miscellaneous

- Series of seminal papers which are part of the FITS standard
- conversion of pixel coordinates to world coordinates is regarded as a multi-step process
- standardized set up keywords to describe coordinate (frames)

Coordinates group

```
/-- COORDINATES_GROUP          Group
    |-- GROUPTYPE              Attr.     string
    |-- REF_LOCATION_VALUE     Attr.     array<double,1>
    |-- REF_LOCATION_UNIT      Attr.     array<string,1>
    |-- REF_LOCATION_FRAME     Attr.     array<string,1>
    |-- REF_TIME_VALUE         Attr.     double
    |-- REF_TIME_UNIT          Attr.     string
    |-- REF_TIME_FRAME         Attr.     string
    |-- NOF_COORDINATES        Attr.     int
    |-- NOF_AXES               Attr.     int
    |-- COORDINATE_TYPES       Attr.     array<string,1>
    |-- COORDINATE0            Group
    |    ...
    `-- COORDINATE{N}          Group
```

- container to collect set of coordinates
  - Direction
  - Linear
  - Tabular
  - Stokes
  - Frequency

Coordinates group

```
/-- COORDINATES_GROUP            Group
    |-- GROUPTYPE                Attr.     string
    |-- REF_LOCATION_VALUE       Attr.     array<double,1>
    |-- REF_LOCATION_UNIT        Attr.     array<string,1>
    |-- REF_LOCATION_FRAME       Attr.     array<string,1>
    |-- REF_TIME_VALUE           Attr.     double
    |-- REF_TIME_UNIT            Attr.     string
    |-- REF_TIME_FRAME           Attr.     string
    |-- NOF_COORDINATES          Attr.     int
    |-- NOF_AXES                 Attr.     int
    |-- COORDINATE_TYPES         Attr.     array<string,1>
    |-- COORDINATE0              Group
    |    ...
    `-- COORDINATE{N}            Group
```

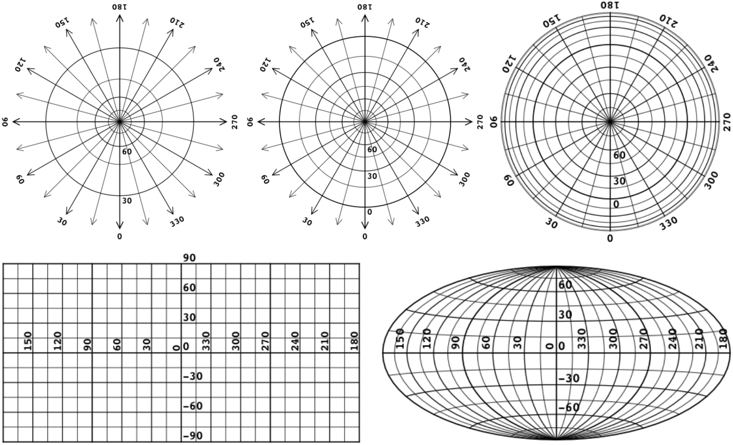
- container to collect set of coordinates
  - Direction
  - Linear
  - Tabular
  - Stokes
  - Frequency
- hold basic data defining common reference frame

Direction Coordinate

Field/Keyword	Type	Description
GROUPTYPE	string	Group type descriptor, DirectionCoord
COORDINATE_TYPE	string	Coordinate Type descriptor, Direction
NOF_AXES	int	N of coordinate axes
AXIS_NAMES	array<string,1>	World axis names
AXIS_UNITS	array<string,1>	Physical units along each coordinate axis.
REFERENCE_VALUE	array<double,1>	Coordinate value at the reference point
REFERENCE_PIXEL	array<double,1>	Array location of the reference point in pixels.
INCREMENT	array<double,1>	Coordinate increment at reference point.
PC	array<double,1>	Linear transformation matrix
EQUINOX	string	Equinox of the observation
RADEC_SYS	string	System of equatorial coordinates
PROJECTION	string	Spherical map projection
PROJECTION_PARAM	array<double,1>	Spherical projection parameters
LONPOLE	double	Native longitude of the celestial pole, ϕ_p
LATPOLE	double	Native latitude of the celestial pole, θ_p.
CONVERSION_SYSTEM	string	Coordinate conversion reference system

- projection of spherical coordinates onto 2-dim space
- specification of algorithm

Spherical map projections



- TAN (Gnomonic), STG (Stereographic), ZEA (Zenithal equal-area)
- CAR (Plate carrée), AIT (Hammer-Aitoff)

```
.                           Group      ---
|-- GROUPTYPE               Attr.      string              'Coordinates'
|-- REF_LOCATION_VALUE      Attr.      array<double,1>
|-- REF_LOCATION_UNIT       Attr.      array<string,1>
|-- REF_LOCATION_FRAME      Attr.      string
|-- NOF_COORDINATES         Attr.      int                 2
|-- NOF_AXES                Attr.      int                 2
|-- COORDINATE_TYPES        Attr.      array<string,1>     ['Linear','Linear']
|-- COORDINATE0             Group      ---
|   |-- GROUPTYPE           Attr.      string              'LinearCoord'
|   |-- COORDINATE_TYPE     Attr.      string              'Linear'
|   |-- NOF_AXES            Attr.      int                 1
|   |-- AXIS_NAMES          Attr.      array<string,1>     ['Time']
|   |-- AXIS_UNITS          Attr.      array<string,1>     ['s']
|   |-- REFERENCE_VALUE     Attr.      array<double,1>     [1.0]
|   |-- REFERENCE_PIXEL     Attr.      array<double,1>     [0.0]
|   |-- INCREMENT           Attr.      array<double,1>     [0.5]
|   `-- PC                  Attr.      array<double,1>     [1.0]
`-- COORDINATE1             Group      ---
    |-- GROUPTYPE           Attr.      string              'LinearCoord'
    |-- COORDINATE_TYPE     Attr.      string              'Linear'
    |-- NOF_AXES            Attr.      int                 1
    |-- AXIS_NAMES          Attr.      array<string,1>     ['Frequency']
    |-- AXIS_UNITS          Attr.      array<string,1>     ['Hz']
    |-- REFERENCE_VALUE     Attr.      array<double,1>     [200.0]
    |-- REFERENCE_PIXEL     Attr.      array<double,1>     [0.0]
    |-- INCREMENT           Attr.      array<double,1>     [10.0]
    `-- PC                  Attr.      array<double,1>     [1.0]
```

```
.                           Group      ---
|-- GROUPTYPE               Attr.      string             'Coordinates'
|-- REF_LOCATION_VALUE      Attr.      array<double,1>
|-- REF_LOCATION_UNIT       Attr.      array<string,1>
|-- REF_LOCATION_FRAME      Attr.      string
|-- NOF_COORDINATES         Attr.      int                2
|-- NOF_AXES                Attr.      int                1
|-- COORDINATE_TYPES        Attr.      array<string,1>    ['Linear','Tabular']
|-- COORDINATE0             Group      ---
|   |-- GROUPTYPE           Attr.      string             'LinearCoord'
|   |-- COORDINATE_TYPE     Attr.      string             'Linear'
|   |-- NOF_AXES            Attr.      int                2
|   |-- AXIS_NAMES          Attr.      array<string,1>    ['x','y']
|   |-- AXIS_UNITS          Attr.      array<string,1>    ['m','m']
|   |-- REFERENCE_VALUE     Attr.      array<double,1>    [0.0,0.0]
|   |-- REFERENCE_PIXEL     Attr.      array<double,1>    [0.0,0.0]
|   |-- INCREMENT           Attr.      array<double,1>    [1.0,1.0]
|   `-- PC                  Attr.      array<double,1>    [1.0,0.0,0.0,1.0]
`-- COORDINATE1             Group      ---
    |-- GROUPTYPE           Attr.      string             'TabularCoord'
    |-- COORDINATE_TYPE     Attr.      string             'Tabular'
    |-- NOF_AXES            Attr.      int                1
    |-- AXIS_NAMES          Attr.      array<string,1>    ['z']
    |-- AXIS_UNITS          Attr.      array<string,1>    ['m']
    |-- PIXEL_VALUES        Attr.      array<double>      [0, 1, 2, 3, ... ]
    `-- WORLD_VALUES        Attr.      array<double>      [0, 1, 4, 9, ... ]
```