SUN/229.2

Particle Physics and Astronomy Research Council
Starlink Project
Starlink User Note 229.2

Tim Jenness, Remo Tilanus,
Horst Meyerdierks, Jon Fairclough
December 2, 2014

The Global Section Datafile (GSD) access library 1.0 Programmer's Manual

---

Abstract

This document describes the Global Section Datafile (GSD) access library. This library provides read-only access to GSD files created at the James Clerk Maxwell Telescope. A description of GSD itself is presented in addition to descriptions of the library routines.

Contents

List of Figures

1 History

The Global Section Datafile (GSD) subroutine library package was written in 1987 by Jon Fairclough [1] to permit the fast reading and writing of data at the James Clerk Maxwell Telescope. Development was stimulated by the need to provide fast filing of data in the so-called "General Single Dish Data" (GSDD) format developed by MRAO, IRAM and NRAO. The JCMT used the GSD format for data storage from all instrumentation until the arrival of SCUBA [2] in 1996 (which uses NDF [3]) and data files in this format will continue to be written by the heterodyne system until the delivery of ACSIS in 2001.

The original GSD I/O library was written in VAX Fortran and has never been ported to a Unix environment. With the move from VMS to Unix in 1994/1995 it was clear that a version of the GSD library was required that would be able to read GSD files (those in the archive as well as new files) without having to change the existing telescope acquisition system or require the use of a VMS application to convert the format on demand. A read-only version of the library was written in C by Remo Tilanus and Horst Meyerdierks in 1994 and was incorporated into the Starlink releases of JCMTDR [4] and SPECX [5].

2 Introduction

The JCMT uses the GSD file format for its current heterodyne acquisition system (via the Dutch Autocorrelation Spectrometer (DAS)) and for its archive of pre-SCUBA data. This document describes the C version of the GSD library (a FORTRAN interface is layered on top), now distributed as a standalone package and not part of an application. The current library cannot be used for creating or modifying GSD files.

3 C interface

The fundamental calling interface is from C and this is documented in appendix C. Routine prototypes can be found in the `gsd.h` include file. An example C program that lists the contents of a GSD file (`gsdprint`) is provided in the distribution.

4 Fortran interface

A Fortran interface is provided that uses the original names of the subroutines rather than C function names. For example, the C routine `gsdOpenRead` should be called from Fortran as `gsd_open_read`. The Fortran binding is incomplete (only covering the supplied C routines) and existing Fortran code may have to be changed before this library can be used. An example routine (`gsd_print.f`) is provided to demonstrate the interface. Additional changes:

- The Fortran include file is now called `GSD_PAR` to fit in with the Starlink naming convention (as opposed to `GSDPARS` in the original VAX library).
- `GSD_PAR` is incomplete. Prior inclusion of `PRM_PAR` is required before `GSD_PAR` can be included.
- Error status values have changed. Zero is good status, non-zero is bad status, but no particular status value can be expected.
- `gsd_inquire_array` is not implemented. Instead `gsd_inq_size` must be used, although it was previously labelled "obsolete".

5 Perl interface

A Perl interface to the GSD library is available, but is not part of this distribution. Please contact Tim Jenness (`t.jenness@jach.hawaii.edu`) for more information.

6 Programming Notes

This section describes some of the basic features of the library in comparison with the VAX version:

- The library provides only read access.
- Only VAX binary GSD files can be read.
- Bad values in the file are converted to PRIMDAT bad values[6], which differ from the traditional VAX/GSD bad values. (This conversion is in memory only, the file itself is unchanged.)
- Type conversion is possible only between numeric types (including logical). Numeric to character or character to numeric conversion is not provided by the library.

7 Programming Tools

The distribution comes with the following programming tools:

gsd_link

Link script used during the link phase to make sure that the correct libraries are used:

```
f77 gsd_print.f -L/star/lib `gsd_link`
```

The math library (`-lm`) is required when using the C interface.

8 Release Notes

This section provides the release notes for the GSD package.

VAX implementation

Implemented in VAX Fortran for the JCMT by Jon Fairclough (1987-1989).

SpecxV6.7

C read-only version released as part of Specx V6.7 in 1995.

V1.0

First version released to Starlink standalone. Unbundled from the SPECX and JCMTDR distributions. First release for Linux.

References

[1] Fairclough J. H., 1989, *GSD – Global Section Datafile System*, JCMT Note MT/IN/33 1, A

[2] Holland W. S., Robson E.I., Gear W.K., Lightfoot J. F., Jenness T., Ivison R. J., Stevens J. A., Cunningham C. R., Ade P. A. R., Griffin M. J., Duncan W. D., Murphy J. A., Naylor D. A., 1999, *MNRAS*, **303**, 659 1

[3] Warren-Smith R. F., 1998, *NDF – Routines for Accessing the Extensible N-Dimensional Data Format*, Starlink User Note 33 1

[4] Lightfoot J. F., Harrison P. A., Meyerdierks H., 1995, *JCMTDR – Applications for reducing JCMT data*, Starlink User Note 132 1

[5] Prestage R. M., Meyerdierks H., Lightfoot J. F., 1995, *SPECX – A millimetre wave spectral reduction package*, Starlink User Note 17 1

[6] Warren-Smith R. F., 1995, *PRIMDAT – processing of primitive numerical data*, Starlink User Note 39 6, A

A Technical Overview

A GSD file has a fairly simple layout. It consists of a 'prolog' followed by 'data'. The prolog describes the data and can be used for retrieving it. The prolog consists of a single "file descriptor" and "item descriptors", one for each data item. The item descriptor locates the required item in the byte stream. More information on the file structure can be found in [1].

The outermost layer of routines is the Fortran binding. This is in `gsd_f77.c`. The routines can only be called from Fortran. They share static external variables amongst themselves (but with no other routines) to record references to up to 100 open GSD files. The calling code only needs

to keep the old file identifier returned by `gsd_open_read`. The routine `gsd_inquire_array` is not implemented, `gsd_inq_size` must be used instead.

The next inner layer is the external C binding. The C binding is similar to the Fortran binding in that there is a one-to-one relationship between routines. The C binding does not use inherited status, but returns a status as the function value. Also, given scalar arguments are passed by value, not by reference. An open GSD file is identfied by no less than four pointers, all of which must be kept by the calling code. The C binding consists of `gsdOpenRead.c`, `gsdClose.c`, `gsdFind.c`, `gsdItem.c`, `gsdInqSize.c`, `gsdGet0x.c`, `gsdGet1x.c`.

The next inner layer contains the `gsd1_` routines. There are three routines used by `gsdOpenRead` to open the file and read its contents into memory. The fourth routine `gsd1_getval` returns information about and values of items to the caller. It retrieves this information from the memory copy of the file as created by `gsdOpenRead`.

The innermost layer are the `gsd2_` routines. There are the `gsd2_nativx` routines, which are used by the `gsd1_` routines. They convert VAX binary file contents to equivalent numbers in the format of the local machine. They also convert VAX/GSD bad values to local/PRIMDAT bad values[6]. Then there is the `gsd2_copya` routine, which is used by `gsd1_getval` to convert from the data type as copied from the file to the data type as required by the calling routine.

B Subroutine List

gsdClose

Close a GSD file

gsdFind

Find GSD item by name

gsdGet0x

Get a scalar value from a GSD file

gsdGet1x

Get an array from a GSD file

gsdInqSize

Inquire array size

gsdItem

Get GSD item by number

gsdOpenRead

Open a GSD file for reading and map it

C Routine Descriptions

This section describes the library interface available to C programmers. The Fortran interface is similar, except the routine names are of the form `GSD_XXX` rather than `gsdXxx`.

gsdClose
Close a GSD file

Description:

This routine closes a GSD file opened previously with `gsdOpenRead`. It also releases the memory that `gsdOpenRead` allocated in connection to that file. For this purpose this routine must be given the standard C file pointer, the pointer to the GSD file descriptor, the pointer to the GSD item descriptors, and the pointer to the data buffer.

Invocation:

```
int gsdClose( FILE *fptr, void *file_dsc, void *item_dsc, char *data_ptr);
```

Arguments:

FILE *fptr (Given)
The file descriptor for the GSD file to be closed.

void *file_dsc (Given)
The GSD file descriptor related to the file opened on fptr.

void *item_dsc (Given)
The array of GSD item descriptors related to the file opened on fptr.

char *data_ptr (Given)
The buffer with all the data from the GSD file opened on fptr.

Returned Value:

int gsdClose();
Status from fclose.

Prototype :

available via #include "gsd.h"

Copyright :

gsdFind
Find GSD item by name

Description:

This routine looks up the GSD item specified by its name and returns the number of the item. This routine also returns the unit string, the type specification and the array flag.

Invocation:

```
 int gsdFind( void *file_dsc, void *item_dsc, char *name, int *itemno, char *unit,
char *type, char *array );
```

Arguments:

void *file_dsc (Given)

The GSD file descriptor related to the file opened on fptr.

void *item_dsc (Given)

The array of GSD item descriptors related to the file opened on fptr.

char *data_ptr (Given)

The buffer with all the data from the GSD file opened on fptr.

char *name (Given)

The name of the item. This should be an array of 16 characters (char name[16]) and a null-terminated string.

int *itemno (Returned)

The number of the item in the GSD file.

char *unit (Returned)

The unit of the item. This should be an array of 11 characters (char name[11]) and will be a null-terminated string.

char *type (Returned)

The data type of the item. This is a single character and one of B, L, W, I, R, D, C.

char *array (Returned)

The array flag. This is a single character and true (false) if the item is (is not) and array.

Returned Value:

int gsdFind();

Status.

- [1:] If the named item cannot be found.
- [0:] Otherwise.

Prototype :

available via #include "gsd.h"

gsdGet0x
Get a scalar value from a GSD file

Description:

This routine returns the value of a scalar GSD item. The item must be specified by the file desciptor, item descriptor array, data array and item number.

<t>	<type>	Fortran	GSD
b	char	byte	byte
l	char	logical∗1	logical
w	short	integer∗2	word
i	int	integer∗4	integer
r	float	real∗4	real
d	double	real∗8	double
c	char[17]	character∗16	char

This routine will convert between numeric types (all but GSD type char). That is to say, the calling routine can request, say, an integer value by calling `gsdGet0i`, even if the item in the GSD file has a different numeric type, say real. C casting rules are applied, which may differ from Fortran truncation rules. No test for conversion errors is performed.

Invocation:

```
 int gsdGet0{blwirdc}( void *file_dsc, void *item_dsc, char *data_ptr, int itemno,
<type> *value );
```

Arguments:

void ∗file_dsc (Given)
The GSD file descriptor.

void ∗item_dsc (Given)
The array of GSD item descriptors related to the GSD file.

char ∗data_ptr (Given)
The buffer with all the data from the GSD file.

int itemno (Given)
The number of the item in the GSD file.

<type> ∗value (Returned)
The data value. For `gsdGet0c` value should be declared with length 17 at least. The returned string is null-terminated in value[16].

Returned Value:

int gsdGet0<t>();

Status.

- [1:] Failure to read the item value.
- [2:] Numbered item cannot be found.
- [3:] Item is not scalar.
- [0:] Otherwise.

Prototype :

available via #include "gsd.h"

Copyright :

gsdGet1x
Get an array from a GSD file

Description:

This routine returns the value of a scalar GSD item. The item must be specified by the file desciptor, item descriptor array, data array and item number.

<t>	<type>	Fortran	GSD
b	char	byte	byte
l	char	logical∗1	logical
w	short	integer∗2	word
i	int	integer∗4	integer
r	float	real∗4	real
d	double	real∗8	double
c	char[16]	character∗16	char

This routine does not convert between types. If the type of the GSD item does not match the type of the routine, then it returns with an error.

It is possible to get only part of the array. Although the part can be specified in terms of an N-dimensional array, this routine does not take a proper N-D section of the array. The caller can specify the start pixel in N dimensions and the end pixel in N dimensions. These two pixels will be converted to memory locations and all memory between the two is returned. This emulates the old GSD library. It is useful really only for parts of 1-D arrays, parts of rows, or single pixels.

Invocation:

```
int gsdGet1{blwird}( void *file_dsc, void *item_dsc, char *data_ptr, int itemno,
int ndims, int *dimvals, int *start, int *end, <type> *values, int *actvals
);
```

Arguments:

void *file_dsc (Given)
The GSD file descriptor.

void *item_dsc (Given)
The array of GSD item descriptors related to the GSD file.

char *data_ptr (Given)
The buffer with all the data from the GSD file.

int itemno (Given)
The number of the item in the GSD file.

int ndims (Given)
The dimensionality the calling routine uses to specify the start and end elements.

int *dimvals (Given)
The array of ndims dimensions (array sizes along each axis).

int *start (Given)
The array indices for the first element.

int *end
The array indices for the last element.

<type> *value (Returned)
The data values. The calling routine must make sure that sufficient memory is provided. Thus it must find out the data type and array size before calling this routine. If the data type is character, then the routine returns a byte buffer with all strings concatenated. There are no string terminators in the buffer and there is none at the end. Each string is 16 byte long and immediately followed by the next string.

int *actvals (Returned)
The number of array values returned. This saves the caller to work out how many array elements correspond to start and end given the dimvals.

Returned Value:

int gsdGet1<t>();
Status.
- [1:] Failure to read the item values.
- [2:] Numbered item cannot be found.
- [4:] Given start and end are inconsistent.
- [0:] Otherwise.

Prototype :

available via #include "gsd.h"

Copyright :

gsdInqSize
Inquire array size

Description:

This routine returns information about the specified array. Returned are the names and units of each dimension, the size along each dimension, and the overall size.

Invocation:

```
 int gsdInqSize( void *file_dsc, void *item_dsc, char *data_ptr, int itemno,
int maxdims, char **dimnames, char **dimunits, int *dimvals, int *actdims, int
*size );
```

Arguments:

void *file_dsc (Given)

The GSD file descriptor.

void *item_dsc (Given)

The array of GSD item descriptors related to the GSD file.

char *data_ptr (Given)

The buffer with all the data from the GSD file.

int itemno (Given)

The number of the item in the GSD file.

int maxdims (Given)

The number of dimensions required and accommodated by the calling routine.

char **dimnames (Returned)

The names for each dimension. The calling routine must provide maxdims pointers to strings. It must also provide the space for the strings, 16 bytes. See Notes for how to declare and pass dimnames.

char **dimunits (Returned)

The units for each dimension. The calling routine must provide maxdims pointers to strings. It must also provide the space for the strings, 11 bytes. See Notes for how to declare and pass dimunits.

int *dimvals (Returned)

The values for each dimension. The calling routine must provide an array of maxdims integers. This would probably be declared as int dimvals[MAXDIMS];

int *actdims (Returned)

The actual number of dimensions. If actdims is less than maxdims, then only actims elements are returned in dimnames, dimunits, dimvals. Further elements declared by the caller are unchanged by this routine.

int *size (Returned)
The total number of elements in the array.

Returned Value:

int gsdInqSize();
Status.

- [1:] Failed to get a dimension value and name.
- [2:] Numbered item does not exist.
- [3:] Array has more dimensions than accommodated by calling routine.
- [0:] Otherwise.

Prototype :

available via #include "gsd.h"

Note :

The calling routine will probably allocate storage for dimension names by declaring a two-dimensional array. That is not suitable for passing to this routine though. The pointers to each string must be copied into an array of pointers. For example:

```
char actual_space[MAXDIMS][16];
char *pointr_array[MAXDIMS];
for ( i = 0; i < MAXDIMS; i++ ) pointr_array[i] = actual_space[i];
status = gsdInqSize( ..., pointr_array, ...  );
```

The reason why this call works but passing actual_space does not work, is that `gsdInqSize` uses the given value as a char **. So in this routine given[1] goes forward in memory by the size of a char * or the number of bytes needed to store a pointer. actual_space would need a step in memory by 16 bytes, i.e. the distance from one string to the next. The main routine knows about this, because it declared actual_space _and_ pointr_array.

Copyright :

gsdItem
Get GSD item by number

Description:

This routine looks up the GSD item specified by its number and returns the name of the item. This routine also returns the unit string, the type specification and the array flag.

Invocation:

```
 int gsdItem( void *file_dsc, void *item_dsc, int itemno, char *name, char *unit,
char *type, char *array );
```

Arguments:

void *file_dsc (Given)

The GSD file descriptor related to the file opened on fptr.

void *item_dsc (Given)

The array of GSD item descriptors related to the file opened on fptr.

char *data_ptr (Given)

The buffer with all the data from the GSD file opened on fptr.

int itemno (Given)

The number of the item in the GSD file.

char *name (Returned)

The name of the item. This should be an array of 16 characters (char name[16]) and will be a null-terminated string.

char *unit (Returned)

The unit of the item. This should be an array of 11 characters (char name[11]) and will be a null-terminated string.

char *type (Returned)

The data type of the item. This is a single character and one of B, L, W, I, R, D, C.

char *array (Returned)

The array flag. This is a single character and true (false) if the item is (is not) and array.

Returned Value:

int gsdFind();

Status.

- [1:] If the named item cannot be found.
- [0:] Otherwise.

Prototype :

available via #include "gsd.h"

gsdOpenRead
Open a GSD file for reading and map it

Description:

This routine opens the named GSD file and reads its contents into memory. It returns a standard C file descriptor, a GSD file descriptor, a pointer to the array of GSD item descriptors, and a pointer to the collective data.

This routine allocates memory to accommodate the GSD file descriptor, the GSD item descriptors, and the data from the GSD file. It also leaves the GSD file open. Any call to this routine must be matched with a call to `gsdClose` with the information returned by this routine. `gsdClose` will close the file and release the memory allocated by this routine.

Invocation:

```
int gsdOpenRead( char *file, float *version, char *label, int *no_items, FILE
**fptr, void **file_dsc, void **item_dsc, char **data_ptr );
```

Arguments:

char *file (Given)
The name of the GSD file to be opened.

float *version (Returned)
The GSD file version number.

char *label (Returned)
The GSD file label. This is a null-terminated string. It should be declared by the calling routine with length 41.

int *no_items (Returned)
The number of items in the GSD file.

FILE **fptr (Returned)
The file descriptor for the GSD file opened.

void **file_dsc (Returned)
The GSD file descriptor. This routine allocates the memory necessary and fills it with the relevant information from the GSD file. A call to `gsdClose` will release this memory (given the pointer).

void **item_dsc (Returned)
The array of GSD item descriptors. This routine allocates the memory necessary and fills it with the relevant information from the GSD file. A call to `gsdClose` will release this memory (given the pointer). The number of array elements is returned in no_items.

char **data_ptr (Returned)
The buffer with all the data from the GSD file. This routine allocates the memory necessary and reads the data into it. A call to `gsdClose` will release this memory (given the pointer). The size of this buffer does not matter, but it can be calculated in bytes as

file_dsc->end_data - file_dsc->str_data + 1 if you know what a struct file_descriptor looks like.

Returned Value:

int gsdOpenRead();

Status. Status is set to

- [1:] Failure to open named file,
- [2:] Failure to read file_dsc from file,
- [3:] Failure to allocate memory for item_dsc,
- [4:] Failure to read item_dsc from file,
- [6:] Failure to read data_ptr from file,
- [7:] Failure to allocate memory for data_ptr,
- [0:] Otherwise.

Prototype :

available via #include "gsd.h"

Copyright :

D Mapping

The following tables define the mapping (in version 5.3 of the JCMT storage task) from item names that will be found in the data files (the "NRAO" name) to the JCMT name. Also included is the nominal FITS header equivalent, emphasis indicating a JCMT-specific variant, and a text description of the field.

Table 1: Mapping of GSD names to FITS equivalents.

NRAO	JCMT	FITS	Description
CELL_V2Y	CELL_V2Y	*YPOSANG*	Position angle of cell y axis (CCW)
UAZ	SDIS(36)	UXPNT	User az correction
UEL	SDIS(37)	UYPNT	User el correction
C1BKE	BACKEND	BACKEND	Name of the backend
C1BTYP	BE_TYPE	*BACKTYPE*	Type of backend
C1DP	DATA_PRECISION	PRECIS	Precision of the data from the backend
C1FTYP	FE_TYPE	*FRONTYPE*	Type of frontend
C1HGT	TEL_HEIGHT	*HEIGHT*	Height of telescope above sea level
C1IFS	IF_DEVICE	IFDEVICE	Name of the IF device
C1LAT	TEL_LATITUDE	*LATITUDE*	Geodetic latitude of telescope (North +ve)
C1LONG	TEL_LONGITUDE	*LONGITUD*	Geographical longitude of telescope (West +ve)
C1OBS	PROJECT_OBS_1	OBSID	Name of the primary observer
C1ONA	PROJECT_OBS_1	OBSERVER	Name of the primary observer
C1ONA1	PROJECT_OBS_2	OBSERVER	Name of the support scientist
C1ONA2	PROJECT_OBS_3	*OPERATOR*	Name of the telescope operator
C1PID	PROJECT	PROJID	Identifies the observing program
C1RCV	FRONTEND	FRONTEND	Name of the frontend
C1SNA1	CENTRE_NAME_1	OBJECT	Source name part 1
C1SNA2	CENTRE_NAME_2	*OBJECT2*	Source name part 2 or altern. name
C1SNO	NOBS	SCAN	Observation number
C1STC	OBS_TYPE	OBSTYPE	Type of observation
C1TEL	TEL_NAME	TELESCOP	Telescope name

Table 1: Mapping of GSD names to FITS equivalents.

NRAO	JCMT	FITS	Description
C2FL	DY	FOCUSL	Secondary mirror y displacement from nominal at observation start
C2FR	DZ	FOCUSR	Secondary mirror z displacement from nominal at observation start
C2FV	DX	FOCUSV	Secondary mirror x displacement from nominal at observation start
C2ORI	SECONDARY_ORI	ORIENT	Rotation or polarization angle orientation of the frontend/reflector
C2PC1	TEL_PC_LAN	PTCON1	Angle by which lower axis is north of ideal
C2PC2	TEL_PC_LAE	PTCON2	Angle by which lower axis is east of ideal
C2PC3	TEL_PC_UANP	PTCON3	Angle by which upper axis is not perpendicular to lower
C2PC4	TEL_PC_BNP	PTCON4	Angle by which beam is not perpendicular to upper axis
C2XPC	TEL_PC_LAZ	AXPOINT	Azimuth/RA enc.zero; enc.reading = az + pc_laz
C2YPC	TEL_PC_UAZ	AYPOINT	Altitude/DEC enc.zero; enc.reading = az + pc_uaz
C3BEFENULO	BES_FE_NULO	*BEFENULO*	Copy of frontend LO frequency per backend section
C3BEFESB	BES_FE_SB_SIGN	*BEFESB*	Copy of frontend sideband sign per backend section
C3BEINCON	BE_IN_CONN	*BEINCON*	IF output channels connected to BE input channels
C3BESCONN	BES_CONN	*BESCON*	BE input channels connected to this section
C3BESSPEC	BES_SPECTRUM	*BESSPEC*	Subsystem nr to which each backend section belongs.
C3BETOTIF	BES_TOT_IF	*BETOTIF*	Total IF per backend section
C3CAL	OBS_CALIBRATION	*OBSCAL*	Calibration observation?
C3CEN	OBS_CENTRE	*OBSCEN*	Centre moves between scans?
C3CL	CYCLE_TIME	CYCLLEN	Duration of each cycle
C3CONFIGNR	DAS_CONF_NR	*CONFIG*	Backend configuration
C3DASCALSRC	DAS_CAL_SOURCE	*CALSRC*	DAS calibration source for backend calibration (POWER or DATA)
C3DASOUTPUT	DAS_OUTPUT	*OUTPUT*	Description of output in DAS DATA (SPECTRUM, T_REC, T_SYS, etc.)
C3DASSHFTFRAC	DAS_SHIFT_FRAC	*SHFTFRAC*	DAS calibration source for backend calibration (POWER or DATA)

Table 1: Mapping of GSD names to FITS equivalents.

NRAO	JCMT	FITS	Description
C3DAT	OBS_UT1D	UTDATE	UT1 date of observation
C3FLY	OBS_CONTINUOUS	*OBSFLY*	Data taken on the fly or in discrete mode?
C3FOCUS	OBS_FOCUS	*OBSFOCUS*	Focus observation?
C3INTT	INTGRN_TIME	*INTGR*	Scan integration time
C3LSPC	NO_BES_O_CH	*NOBESDCH*	Number of channels per backend section
C3LST	OBS_LST	LST	Local sidereal time at the start of the observation
C3MAP	OBS_MAP	*OBSMAP*	Map observation?
C3MXP	NO_SCAN_PNTS	*NOSCNPTS*	Maximum number of map points done in a phase
C3NCH	NO_BE_O_CH	*NOBEOCH*	No.backend output channels
C3NCI	NO_CYCLES	*NOCYCLES*	Maximum number of cycles in the scan
C3NCP	NO_CYCLE_PNTS	*NOCYCPTS*	Total number of xy positions observed during a cycle
C3NCYCLE	NCYCLE	*NCYCLE*	Number of cycles done in the scan
C3NFOC	NO_FE_O_CH	*NOFCHAN*	NO_FE_O_CH:No. of frontend output channels
C3NIS	NO_SCANS	*NOSCANS*	Number of scans
C3NLOOPS	NO_SCANS	NSCANS	Number of scans per observation commanded at observation start
C3NMAP	NO_MAP_PNTS	NOPTS	Number of map points
C3NOIFPBES	NO_IF_PER_BES	*NOIFPBES*	Number of IF inputs to each section (2 for correlator, 1 for AOS)
C3NO_SCAN_VARS1	NO_SCAN_VARS1	*NOSCNVR1*	Number of scan table 1 variables
C3NO_SCAN_VARS2	NO_SCAN_VARS2	*NOSCNVR2*	Number of scan table 2 variables
C3NPP	NO_MAP_DIMS	*NOMAPDIM*	Number of dimension in the map table
C3NRC	NRC	NORCHAN	NO_BE_I_CH:No.backend input channels
C3NRS	NO_BES	*NORSECT*	Number of backend sections
C3NSAMPLE	NSCAN	*NSCAN*	Number of scans done

Table 1: Mapping of GSD names to FITS equivalents.

NRAO	JCMT	FITS	Description
C3NSV	NO_PHASE_VARS	NOSWVAR	Number of phase table variables
C3OVERLAP	BES_OVERLAP	*OVERLAP*	Subband overlap
C3PPC	NO_PHASES	NOPHASE	Number of phases per cycle
C3SRT	SCAN_TIME	SAMPRAT	Total time of scan (=total integration time if OBS_CONTINUOUS = .FALSE.)
C3UT	OBS_UT1H	UT	UT1 hour of observation
C3UT1C	OBS_UT1C	*UT1C*	UT1-UTC correction interpolated from time service telex (in days)
C4AMPL_EW	AMPL_EW	CHOPAZ	Secondary mirror chopping amplitude parallel to lower axis
C4AMPL_NS	AMPL_NS	CHOPEL	Secondary mirror chopping amplitude parallel to upper axis
C4AXY	CELL_X2Y	*XYANGLE*	Angle between cell y axis and x-axis (CCW)
C4AZ	CENTRE_AZ	AZ	Azimuth at observation date
C4AZERR	SDIS(7)	XPOINT	DAZ:Net Az offset at start (inc.tracker ball setting and user correction)
C4CECO	CENTRE_CODE	*COORDCD2*	centre coords. AZ= 1;EQ=3;RD=4;RB=6;RJ=7;GA=8
C4CSC	CENTRE_COORDS	COORDCD	Character code of commanded centre or source coordinate system
C4DAZ	DAZ_SM	XPOINT	Telescope lower axis correction for secondary mirror XYZ
C4DECDATE	CENTRE_DEC	*DECDATE*	Declination of date
C4DEL	DEL_SM	YPOINT	Telescope upper axis correction for secondary mirror XYZ
C4DO1	CELL_X	DESORG1	Cell x dimension; descriptive origin item 1
C4DO2	CELL_Y	DESORG2	Cell y dimension; descriptive origin item 2
C4DO3	CELL_V2X	DESORG3	Angle by which the cell x axis is oriented with respect to local vertical
C4EDC	CENTRE_DEC	EPOCDEC	Declination of date
C4EDEC	CENTRE_DEC1950	EPOCDEC	Declination of source for EPOCH
C4EDEC2000	CENTRE_DEC2000	*DECJ2000*	Declination J2000
C4EL	CENTRE_EL	EL	Elevation at observation date

Table 1: Mapping of GSD names to FITS equivalents.

NRAO	JCMT	FITS	Description
C4ELERR	SDIS(8)	YPOINT	DEL:Net El offset at start (inc.tracker ball setting and user correction)
C4EPH	CENTRE_EPOCH	EPOCH	Date of the RA/DEC coordinates (1950)
C4EPT	EPOCH_TYPE	*EPOCHTYP*	Type of epoch, JULIAN, BESSELIAN or APPARENT
C4ERA	CENTRE_RA1950	EPOCRA	Right ascension of source for EPOCH
C4EW_ENCODER	AMPL_E_SET	*EW_ENCOD*	Secondary mirror ew encoder value
C4EW_SCALE	EW_AMPL_SCALE	*EW_SCALE*	Secondary mirror ew chop scale
C4FRQ	FREQUENCY	*CHOPFREQ*	Secondary mirror chopping period
C4FUN	WAVEFORM	*WAVEFORM*	Secondary mirror chopping waveform
C4GB	CENTRE_GB	GALLAT	Galactic latitude
C4GL	CENTRE_GL	GALLONG	Galactic longitude
C4LSC	CELL_COORDS	FRAME	Char. code for local x-y coord.system
C4MCF	CENTRE_MOVING	*CENMOVE*	Centre moving flag (solar system object)
C4MOCO	TEL_COORDS	*MOUNTING*	Mounting of telescope; defined as LOWER/UPPER axes, e.g; AZ/ALT
C4NS_ENCODER	AMPL_N_SET	*NS_ENCOD*	Secondary mirror ns encoder value
C4NS_SCALE	NS_AMPL_SCALE	*NS_SCALE*	Secondary mirror ns chop scale
C4ODCO	CELL_UNIT	*MAPUNIT*	Units of cell and mapping coordinates;offset definition code
C4OFFS_EW	OFFS_EW	*EWTILT*	Secondary mirror offset parallel to lower axis (East-West Tilt)
C4OFFS_NS	OFFS_NS	*NSTILT*	Secondary mirror offset parallel to upper axis (North-South Tilt)
C4PER	PERIOD	CHOPTIME	Secondary mirror chopping period
C4POSANG	POSANG	*CHOPDIRN*	Secondary mirror chop position angle
C4RA2000	CENTRE_RA2000	*RAJ2000*	Right ascension J2000
C4RADATE	CENTRE_RA	*RADATE*	Right Ascension of date
C4RX	REFERENCE_X	*XREF*	Reference x position (JCMT cells wrt to centre; NRAO abs. degrees)

Table 1: Mapping of GSD names to FITS equivalents.

NRAO	JCMT	FITS	Description
C4RY	REFERENCE_Y	*YREF*	Reference y position (JCMT cells wrt to centre; NRAO abs. degrees)
C4SM	CHOPPING	*CHOPPING*	Secondary mirror is chopping
C4SMCO	COORDS	*CHOPCOOR*	Secondary mirror chopping coordinate system
C4SX	CENTRE_OFFSET_X	*XSOURCE*	Commanded x centre position (JCMT cells wrt to centre; NRAO abs. degrees)
C4SY	CENTRE_OFFSET_Y	*YSOURCE*	Commanded y centre position (JCMT cells wrt to centre; NRAO abs. degrees)
C4THROW	THROW	*CHOPTHRW*	Secondary mirror chop throw
C4X	X	*AFOCUSV*	Secondary mirror absolute X position at observation start
C4Y	Y	*AFOCUSH*	Secondary mirror absolute Y position at observation start
C4Z	Z	*AFOCUSR*	Secondary mirror absolute Z position at observation start
C5AT	TAMB	TAMB	Ambient temperature
C5DP	DEW_POINT	DEW_PT	Mean atmospheric dew point
C5IR	REF_INDEX	REFRAC	Mean atmospheric refractive index (alternative)
C5IR1	TEL_REFR_A	REFRAC1	Refraction constant A (see MTIN026)
C5IR2	TEL_REFR_B	REFRAC2	Refraction constant B (see MTIN026)
C5IR3	TEL_REFR_C	REFRAC3	Refraction constant C (see MTIN026)
C5MM	VAP_PRESSURE	MMH2O	Mean atmospheric vapour pressure
C5PRS	PAMB	PRESSURE	Mean atmospheric pressure
C5RH	HAMB	HUMIDITY	Mean atmospheric relative humidity
C6CYCLREV	CYCLE_REVERSAL	*CYCLREV*	Cycle reversal flag
C6DX	CELL_X	DELTAX	Cell x dim,; descriptive origin item 1
C6DY	CELL_Y	DELTAY	Cell y dimension; descriptive origin item 2
C6FC	CELL_CODE	*FRAME2*	Local x-y AZ= 1;EQ=3;RD=4;RB=6;RJ=7;GA=8
C6MODE	SWITCH_MODE	*SWMODE*	Observation mode

Table 1: Mapping of GSD names to FITS equivalents.

NRAO	JCMT	FITS	Description
C6MSA	CELL_V2X	SCANANG	Scanning angle - angle from local vertical to x axis measured CW
C6NP	NCYCLE_PNTS	*NCYCPTS*	Number of sky points completed in the observation
C6REV	SCAN_REVERSAL	*REVERSAL*	Map rows scanned in alternate directions?
C6SD	OBS_DIRECTION	DIRECTN	Map rows are in X (horizontal) or Y(vertical) direction
C6ST	OBS_TYPE	OBSMODE	Type of observation
C6XGC	X_MAP_START	XCELL0	X coordinate of the first map point
C6XNP	NO_X_MAP_PNTS	NOXPTS	X map dimension; number of points in the x-direction
C6XPOS	X_MAP_POSITIVE	*XSIGN*	In first row x increases (TRUE) or decreases (FALSE)
C6YGC	Y_MAP_START	YCELL0	Y coordinate of the first map point
C6YNP	NO_Y_MAP_PNTS	NOYPTS	Y map dimension; number of points in the y-direction
C6YPOS	Y_MAP_POSITIVE	*YSIGN*	In first row y increases (TRUE) or decreases (FALSE)
C7AP	APERTURE	*APERTURE*	Aperture
C7BCV	BAD_CHANNEL	BADCHV	Bad channel value
C7CAL	CAL_TYPE	TYPECAL	Calibration type (standard or direct, chopperwheel)
C7FIL	FILTER	*FILTER*	Filter
C7HP	FWHM		FWHM of the beam profile (mean)
C7NIF	NO_IF_CH	NOIFCHAN	Number of IF channels
C7OSN	OBS_REF_SCAN	OFFSCAN	Calibration observation number (in which a standard was observed)
C7PHASE	PHASE	*PHASE*	Lockin phase
C7SEEING	SAO_SEEING	*SEEING*	Seeing at JCMT
C7SEETIME	SAO_YYMMDDHHMM	*SEETIME*	SAO seeing time (YYMMDDHHMM)
C7SNSTVTY	SENSITIVITY	*SNSTVTY*	Lockin sensitivity in scale range units
C7SNTVTYRG	RANGE	*SNTVTYRG*	Sensitivity range of lockin

Table 1: Mapping of GSD names to FITS equivalents.

NRAO	JCMT	FITS	Description
C7SZVRAD	SZVRAD	SZVRAD	Number of elements of vradial array
C7TAU225	CSO_TAU	*TAU225*	CSO tau at 225GHz
C7TAURMS	CSO_TAU_RMS	*TAURMS*	CSO tau rms
C7TAUTIME	CSO_YYMMDDHHMM	*TAUTIME*	CSO tau time (YYMMDDHHMM)
C7TIMECNST	TIME_CONSTANT	*TIMECNST*	Lockin time constant
C7VC	VEL_COR	RVSYS	Velocity correction
C7VR	VELOCITY	VELOCITY	Radial velocity of the source
C7VRD	VEL_DEF	VELDEF	Velocity definition code; method of computing the velocity
C7VREF	VEL_REF	VELREF	Velocity reference code; reference point for telescope & source velocity
C8AAE	APERTURE_EFF	APPEFF	Ratio total power observed/incident on the telescope
C8ABE	BEAM_EFF	BEAMEFF	Fraction of beam in diffraction limited main beam
C8EF	ETAFSS	ETAFSS	Forward spillover and scattering efficiency
C8EL	ETAL	ETAL	Rear spillover and scattering efficiency
C8GN	ANTENNA_GAIN	ANTGAIN	Antenna gain
C9OT	TEL_TOLERANCE	OBSTOL	Observing tolerance
C11PHA	PHASE_TABLE	*PHASTB*	Phase table: switching scheme dependent
C11VD	PHASE_VARS	*VARDES*	Names of the cols. of phase table
C12ALPHA	BES_ALPHA	*ALPHA*	Ratio of signal sideband to image sideband sky transmission
C12BM	BES_BITMODE	BM	Correlation bit mode
C12BW	BANDWIDTH	BW	Bandwidth
C12CAL	DATA_UNITS	*DATAUNIT*	Units of spectrum data
C12CALTASK	BE_CAL_TASK	*CALTASK*	Calibration instrument used (FE, BE, or USER)
C12CALTYPE	BE_CAL_TYPE	*CALTYPE*	Type of calibration (THREELOADS or TWOLOADS)

Table 1: Mapping of GSD names to FITS equivalents.

NRAO	JCMT	FITS	Description
C12CF	FE_NUOBS	OBSFREQ	Centre frequency
C12CM	BES_CORR_MODE	CM	Correlation function mode
C12CT	TCAL	TCAL	Calibration temperature
C12ETASKY	BES_ETA_SKY	*ETA_SKY*	Sky transmission from last calibration
C12ETASKYIM	BES_ETA_SKY_IM	*ETA_SKY_*	Frontend-derived sky transmission
C12ETATEL	BES_ETA_TEL	*ETA_TEL*	Telescope transmission
C12FR	DELTANU	FREQRES	Frequency resolution [MHz]
C12GAINS	GAIN	*GAINS*	Gains
C12GNORM	G_NORM	GNORM	Data normalisation factor
C12GREC	GREC		Raw data units per Kelvin
C12GS	BES_G_S	*G_S*	Normalizes signal sideband gain
C12INFREQ	BE_NUIN	*INFREQ*	BE input frequencies [GHz]
C12NOI	NOISE	*NOISE*	Noise value
C12REDMODE	BE_RED_MODE	*REDMODE*	Way of calibrating the data (RATIO or DIFFERENCE)
C12RF	FE_NUREST	RESTFREQ	Rest frequency
C12RST	TSYS_OFF	RTSYS	Reference system temperature
C12RT	TREC	TRX	Receiver temperature
C12SBRAT	BE_SB_RATIO	*SBRATIO*	Sideband ratio
C12SCAN_TABLE_1	SCAN_TABLE1	*SCANTAB1*	Begin scan table
C12SCAN_TABLE_2	SCAN_TABLE2	*SCANTAB2*	End scan table
C12SCAN_VARS1	SCAN_VARS1	*SCANVRS1*	Names of the cols. of scan table1
C12SCAN_VARS2	SCAN_VARS2	*SCANVRS2*	Names of the cols. of scan table2
C12SST	TSYS_ON	STSYS	Source system temperature

Table 1: Mapping of GSD names to FITS equivalents.

NRAO	JCMT	FITS	Description
C12TAMB	T_HOT	*THOT*	Ambient load temperature
C12TASKY	BES_TA_SKY	*TA_SKY*	Ratio of signal sideband to image sideband sky transmission
C12TCOLD	T_COLD	*TCOLD*	Cold load temperature
C12TSKY	TSKY	*TSKY*	Sky temperature at last calibration
C12TSKYIM	BES_T_SKY_IM	*T_SKY_IM*	Frontend-derived Tsky, image sideband
C12TSYSIM	BES_T_SYS_IM	*T_SYS_IM*	Frontend-derived Tsys, image sideband
C12TTEL	TTEL	*TTEL*	Telescope temp. from last skydip
C12VCOLD	IF_V_COLD	*VCOLD*	
C12VDEF	VEL_DEFN		Velocity definition code - radio, optical, or relativistic
C12VHOT	IF_V_HOT	*VHOT*	
C12VREF	VEL_REF		Velocity frame of reference - LSR, Bary-, Helio-, or Geo- centric
C12VSKY	IF_V_SKY	*VSKY*	
C12WO	H2O_OPACITY	TAUH2O	Water opacity
C13DAT	DATA	SPECT	Reduced data
C13ERR	ERROR	TRMS	Standard error
C13RAW_ERROR	DATA_ERROR	*RAWERROR*	
C13RAW_ERROR_OP	RAW_ERROR_OP	*RAWERROR*	Raw (out of phase) error also to be stored at end scan
C13RESP	RESP	*RESP*	array of responsivities
C13SPV	SAMPLES	*RAWSPECT*	
C13SPV_OP	SAMPLES_OP	*RAWSPECT*	Raw out of phase data samples in each phase
C13STD	STDEV	*STDSPECT*	Phase data standard deviation
C14PHIST	MAP_TABLE	*MAPTABLE*	List of xy offsets for each scan

Table 1: Mapping of GSD names to FITS equivalents.

NRAO	JCMT	FITS	Description
C90T	TEL_TOLERANCE	OBSTOL	Observing tolerance