



**Robert C. Byrd Green Bank Telescope
NRAO Green Bank**

M. H. Clark

January 30, 2003

GBT SOFTWARE PROJECT NOTE 10.6

GBT IFManager FITS File Specification

HTML version Available¹

Contents

1 Background	3
2 Primary HDU	3
3 IF Binary Table Extension	3

Abstract

The FITS format structure is presented for the GBT IF scan data files. The scan data FITS files are permanently archived after each observation, and will usually be input to the aips++ filler, which has the task of combining all scan data FITS files into an aips++ Measurement Set. This data may then either be processed further in aips++, or written out in an aips++ supported FITS format (one of which is single dish FITS).

The scan data FITS files contain both 1) a complete description of the complete IF signal paths for the observation and 2) key parameters describing the resultant signal at the backend.

¹<http://www.gb.nrao.edu/GBT/MC/doc/dataproc/gbtIFFits/gbtIFFits/gbtIFFits.html>

History

- 8th November 2001** Penultimate definition of FITSVVER 1.1; released for general comment (Richard Prestage).
- 16th November 2001** Final version agreed for implementation of FITSVVER 1.1. Added history. Subsequent changes to the content of this document should increment the document version number (Richard Prestage).
- 4th December 2001** Revamped definition for FITSVVER 1.1: addition of keywords FEED and HIGH_CAL are postponed for FITSVVERS 1.2 (Mark Clark).
- 23rd March 2002** Modified to correctly document that all TFORM characters are preceded by an integer even if *1* (Mark Clark).
- 30th October 2002** Modified to fully document FITSVVERS 1.2, i.e., addition of columns for *SRFEED1* and *SRFEED2* (Mark Clark).
- 30th January 2003** Modified to fully document FITSVVERS 1.3, i.e., use of BANDWIDTH of zero to denote the IF signal is outside the limits of the emulation and modifications to prevent the need of NULLs in such cases as seen in previous versions. (Mark Clark).
- 3rd April 2008** Modified to mention FREQOFF offset found in the STATE table of the LO1 Device FITS file (Bob Garwood).

1 Background

All GBT scan IF FITS files should conform to the standards specified in the GBT Software Project Note 4.0, “Device and Log FITS Files for the GBT.” The RF signal as initially received at the antenna’s receiver undergoes numerous transformations by telescope devices as it is passed from the receiver’s feed to the backend’s input port. The characteristics of a signal at a backend’s input port is a function of the devices, device settings, and cabling along the path. The IFManager contains representations of telescope devices, accepts device settings, and reads a cabling file in order to emulate the expected terminal signal characteristics for all complete paths from a source (receiver or noise diode) to a backend port. The emulation handles changes to frequency characteristics, but not power levels. Each row in the IF FITS file represents one complete path. If the path is being altered by the sig/ref switching signal directly controlling an IF switch then the path description represents the signal position. Information in each row includes initial and terminating points in the path, polarization, nominal center frequencies, test tone characteristics (if any), Center Frequency Formual (CFF) coefficients, and a list of all devices and components affecting the signal.

2 Primary HDU

The IFManager FITS keywords for the primary HDU conforms to the definition for common FITS headers as described in GBT Software Project Note 4.0 “Device and Log FITS Files for the GBT.”

3 *IF* Binary Table Extension

Each row of the *IF* table describes a complete IF path from source to backend. All complete paths for a scan are entered regardless whether the path terminates on an active backend or not.

```
XTENSION= 'BINTABLE'           / binary table extension
BITPIX  =                      8 / 8-bit bytes
NAXIS   =                      2 / 2-dimensional binary table
NAXIS1  =                      4370 / width of table in bytes
NAXIS2  =                      2 / number of rows in table
PCOUNT  =                      0 / size of special data area
GCOUNT  =                      1 / one data group (required keyword)
TFIELDS =                      25 / number of fields in each row
TTYPER1 = 'BACKEND'           / label for field  1
TFORM1  = '32A'               / data format of field: ASCII Character
TUNIT1  = 'none'              / physical unit of field
TTYPER2 = 'BANK'              / label for field  2
TFORM2  = '2A'                / data format of field: ASCII Character
TUNIT2  = 'none'              / physical unit of field
TTYPER3 = 'PORT'              / label for field  3
TFORM3  = '1J'                / data format of field: 4-byte INTEGER
TUNIT3  = 'none'              / physical unit of field
TTYPER4 = 'RECEIVER'         / label for field  4
TFORM4  = '32A'               / data format of field: ASCII Character
TUNIT4  = 'none'              / physical unit of field
TTYPER5 = 'FEED'              / label for field  5
TFORM5  = '1J'                / data format of field: 4-byte INTEGER
TUNIT5  = 'none'              / physical unit of field
TTYPER6 = 'SRFEED1'          / label for field  6
TFORM6  = '1J'                / data format of field: 4-byte INTEGER
```

```

TUNIT6 = 'none      ' / physical unit of field
TTYPER7 = 'SRFEED2  ' / label for field  7
TFORM7  = '1J      ' / data format of field: 4-byte INTEGER
TUNIT7  = 'none      ' / physical unit of field
TTYPER8 = 'RECEPTOR' / label for field  8
TFORM8  = '8A      ' / data format of field: ASCII Character
TUNIT8  = 'none      ' / physical unit of field
TTYPER9 = 'LO_CIRCUIT' / label for field  9
TFORM9  = '32A     ' / data format of field: ASCII Character
TUNIT9  = 'none      ' / physical unit of field
TTYPER10 = 'LO_COMPONENT' / label for field 10
TFORM10 = '32A     ' / data format of field: ASCII Character
TUNIT10 = 'none      ' / physical unit of field
TTYPER11 = 'SIDEBAND' / label for field 11
TFORM11 = '2A      ' / data format of field: ASCII Character
TUNIT11 = 'none      ' / physical unit of field
TTYPER12 = 'POLARIZE' / label for field 12
TFORM12 = '2A      ' / data format of field: ASCII Character
TUNIT12 = 'none      ' / physical unit of field
TTYPER13 = 'CENTER_IF' / label for field 13
TFORM13 = '1E      ' / data format of field: 4-byte REAL
TUNIT13 = 'Hz       ' / physical unit of field
TTYPER14 = 'CENTER_SKY' / label for field 14
TFORM14 = '1E      ' / data format of field: 4-byte REAL
TUNIT14 = 'Hz       ' / physical unit of field
TTYPER15 = 'BANDWIDTH' / label for field 15
TFORM15 = '1E      ' / data format of field: 4-byte REAL
TUNIT15 = 'Hz       ' / physical unit of field
TTYPER16 = 'HIGH_CAL' / label for field 16
TFORM16 = '1J      ' / data format of field: 4-byte INTEGER
TUNIT16 = 'none      ' / physical unit of field
TTYPER17 = 'TEST_TONE_IF' / label for field 17
TFORM17 = '1E      ' / data format of field: 4-byte REAL
TUNIT17 = 'Hz       ' / physical unit of field
TTYPER18 = 'TEST_TONE_SKY' / label for field 18
TFORM18 = '1E      ' / data format of field: 4-byte REAL
TUNIT18 = 'Hz       ' / physical unit of field
TTYPER19 = 'TEST_TONE_CIRCUIT' / label for field 19
TFORM19 = '32A     ' / data format of field: ASCII Character
TUNIT19 = 'none      ' / physical unit of field
TTYPER20 = 'TEST_TONE_COMPONENT' / label for field 20
TFORM20 = '32A     ' / data format of field: ASCII Character
TUNIT20 = 'none      ' / physical unit of field
TTYPER21 = 'SFF_MULTIPLIER' / label for field 21
TFORM21 = '1D      ' / data format of field: 8-byte DOUBLE
TUNIT21 = 'none      ' / physical unit of field
TTYPER22 = 'SFF_SIDEBAND' / label for field 22
TFORM22 = '1D      ' / data format of field: 8-byte DOUBLE
TUNIT22 = 'none      ' / physical unit of field
TTYPER23 = 'SFF_OFFSET' / label for field 23
TFORM23 = '1D      ' / data format of field: 8-byte DOUBLE
TUNIT23 = 'none      ' / physical unit of field
TTYPER24 = 'TRANSFORM_COUNT' / label for field 24
TFORM24 = '1J      ' / data format of field: 4-byte INTEGER

```

```

TUNIT24 = 'none      '           / physical unit of field
TTYTYPE25 = 'TRANSFORMS'        / label for field 25
TFORM25 = '4096A:SSTR256/059'  / data format of field: ASCII Character
TUNIT25 = 'none      '           / physical unit of field
EXTNAME = 'IF          '         / name of this binary table extension
COMMENT   Sky Frequency Formula:
COMMENT   sky = SFF_SIDE BAND*IF + SFF_MULTIPLIER*LO1 + SFF_OFFSET
COMMENT   Signed Sum of the LOs:
COMMENT   sum = -(SFF_MULTIPLIER*LO1 + SFF_OFFSET)/SFF_SIDE BAND
COMMENT   BANDWDTH of 0 denotes the bandpass is outside the optimal range
COMMENT   BACKEND: name of the terminating backend
COMMENT   BANK: name of the backend's set of inputs
COMMENT   PORT: index of the backend's input
COMMENT   RECEIVER: name of the receiver of origin
COMMENT   FEED: index of receiver RF entry point (0 indicates none)
COMMENT   SRFEED1: index of first FEED of a sig/ref pair
COMMENT   SRFEED2: index of second FEED of a sig/ref pair
COMMENT   RECEPTOR: name of the receiver's detector
COMMENT   LO_CIRCUIT: circuit producing the tracking frequency
COMMENT   LO_COMPONENT: component producing the tracking frequency
COMMENT   SIDEBAND: resulting sideband: upper or lower
COMMENT   POLARIZE: resulting polarization
COMMENT   CENTER_IF: approximate physical center frequency
COMMENT   CENTER_SKY: approximate center frequency on the sky
COMMENT   BANDWDTH: approximate resulting bandwidth
COMMENT   HIGH_CAL: 1 indicates a high calibrator was used
COMMENT   TEST_TONE_IF: approximate physical test tone frequency, if any
COMMENT   TEST_TONE_SKY: approximate test tone frequency on the sky, if any
COMMENT   TEST_TONE_CIRCUIT: circuit producing the test tone, if any
COMMENT   TEST_TONE_COMPONENT: component producing the test tone, if any
COMMENT   SFF_MULTIPLIER: Sky Frequency Formula multiplier coefficient
COMMENT   SFF_SIDE BAND: Sky Frequency Formula sideband coefficient
COMMENT   SFF_OFFSET: Sky Frequency Formula offset coefficient
COMMENT   TRANSFORM_COUNT: number of transform
COMMENT   TRANSFORMS: matrix of transform descriptions (frequencies in MHz)
END

```

The first three columns (BACKEND, BANK, and PORT) specify the termination point of a signal path. The same values are used as identifiers in the backend files respectively as INSTRUME, BANK and PORT.

The next seven columns (RECEIVER, FEED, SRFEED1, SRFEED2, RECEPTOR, LO_CIRCUIT and LO_COMPONENT) specify the origin of a signal path. The value of RECEIVER is the same as for INSTRUME in the primary HDU of the calibration FITS files, and the keywords FEED and RECEPTOR are used in the RX_CAL_INFO binary tables of the calibration FITS files. If FEED is part of a sig/ref pair, i.e., it may be used as part of nodding or beamswitching, then the indices of the two FEEDs are given by SRFEED1 and SRFEED2, otherwise they are set to 0. The keywords LO_CIRCUIT and LO_COMPONENT identify the oscillator used for Doppler tracking and/or frequency-switching.

The following six columns (SIDEBAND, POLARIZE, CENTER_IF, BANDWDTH, and HIGH_CAL) describe the signal itself. The values for SIDEBAND can be upper ('U') or lower ('L'). The values for POLARIZE can be unknown ('U'), linear X ('X'), linear Y ('Y'), right circular ('R'), or left circular ('L'). The values of CENTER_IF and CENTER_SKY are the center frequencies of the nominal band pass of the signal for the IF itself and the corresponding sky frequency. The value of BANDWDTH is the nominal band pass. These values are nominal because ideal filters are used in the emulation usually at the 2 dB or half-power points. If the bandpass is outside the nominal limits of the system, then the BANDWDTH is set to zero to denote the configuration is marginal. The

value of the column HIGH_CAL indicates whether the high calibration noise tube was fired during cal phases. Some receivers allow the user to select the calibration level.

The columns TEST_TONE_IF, TEST_TONE_SKY, TEST_TONE_CIRCUIT, TEST_TONE_COMPONENT describe the center frequencies and source of an optional test tone that may be added to the signal path at the receiver.

The next three columns (SFF_MULTIPLIER, SFF_SIDE BAND, and SFF_OFFSET) are coefficients represented as doubles and are used to express the relationship between an IF frequency and its associated sky frequency using either the *Sky Frequency Formula* or the *Signed Sum of the LOs* equations. The other independent variable in the equation is the tracking LO frequency (LO1FREQ) which may vary between switching signal phases and is specified in the LO1 Device FITS file (see GBT Software Project Note 006). There may also be an additional frequency offset that varies with switching phase (STATE) that is added to the sky frequency formula. The value of this offset is given by the FREQOFF column in the STATE table of the LO1 Device FITS file (see GBT Software Project Note 006).

The final two columns (TRANSFORM_COUNT and TRANSFORMS) represent a terse textual description of all the circuits/components the signal passed through and their individual effects. This description is not needed for analysis, but is useful for post mortems since it is a thorough description of the IFManager's emulation.