LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note **LIGO-T040164-v2** 2022/11/12

SFT Data Format Version 2–3 Specification

Bruce Allen, Evan Goetz, David Keitel, Michael Landry, Gregory Mendell, Reinhard Prix, Keith Riles, Karl Wette
LIGO Scientific Collaboration, Continuous Wave Group

California Institute of Technology
LIGO Project, MS 18-34
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project, Room NW17-161
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
Route 10, Mile Marker 2
Richland, WA 99352
Phone (509) 372-8106
Fax (509) 372-8137
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
19100 LIGO Lane
Livingston, LA 70754
Phone (225) 686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

`http://www.ligo.caltech.edu/`

Git 33adc79a 2022-11-12T21:37:24+11:00 — Processed with LaTeX on 2022/11/12

Abstract

This document describes versions 2 and 3 of the SFT (Short Fourier Transform) data format. (Version 1 of the format was never formally described.) Differences between the versions are denoted with the text **(Version 2)** and **(Version 3)** respectively. Version 3 is backward compatible with version 2.

This document also describes conventions for naming SFT files and organizing them into directories. The conventions are optional, but encouraged for interoperability with other tools, and particularly for SFT files which are published and distributed across LSC clusters. The filename convention is backward compatible with previous conventions, except that the character '`#`' is no longer allowed, and the characters '_' and '`+`' are reserved for delimiting fields in the filename.

Contents

1 SFT Data Format Specification

1.1 Specification

An SFT is stored in a file. See Section 2.1 for file name conventions. The file is composed of concurrent SFT BLOCKs. Each SFT BLOCK is organized as follows:

Summary of SFT BLOCK structure

Label	Size
HEADER	48 bytes
ASCII COMMENT	$8n$ bytes, where n is a non-negative integer
DATA	$8N$ bytes, where N is a positive integer

The total length of the SFT BLOCK is $48 + 8n + 8N$ bytes. The SFT BLOCK may be written in either big-endian or little-endian ordering. All floats and doubles follow the IEEE-754 floating-point conventions.

The HEADER contains 48 bytes as follows:

Summary of HEADER structure

Size	C Type	C Structure Field Name
8 bytes	`REAL8`	`version`
4 bytes	`INT4`	`gps_sec`
4 bytes	`INT4`	`gps_nsec`
8 bytes	`REAL8`	`tbase`
4 bytes	`INT4`	`first_frequency_index`
4 bytes	`INT4`	`nsamples`
8 bytes	`UINT8`	`crc64`
2 bytes	`CHAR`	`detector[2]`
2 bytes	`UINT2`	**(Version 2)**: `padding` (unused) **(Version 3)**: `windowspec`
4 bytes	`INT4`	`comment_length`

The SFT blocks in a given SFT file are required to:

1. come from the same instrument, and have:
2. identical version numbers,
3. monotonically increasing GPS times,
4. identical values of `tbase`,
5. identical values of `first_frequency_index`,
6. identical values of `nsamples`,

7. **(Version 3)** identical values of `windowspec`.

Note: SFT blocks in a given SFT file are in general NOT contiguous. In other words the GPS start time of a given block may or may not equal the GPS start time of the previous block plus the time baseline.

Note that the HEADER corresponds to the C structure below:

```
struct SFTtag {
  REAL8 version;
  INT4  gps_sec;
  INT4  gps_nsec;
  REAL8 tbase;
  INT4  first_frequency_index;
  INT4  nsamples;
  UINT8 crc64;
  CHAR  detector[2];
  UINT2 windowspec;           /* (Version 2): unused */
  INT4  comment_length;
} SFTheader;
```

when the structure is packed, i.e. no zero padding between fields is allowed. Note that several of these quantities that might be taken as unsigned are in fact signed. This makes it easier and less error-prone for user applications and code to compute differences between these quantities.

The structure of the ASCII COMMENT is `comment_length` $= 8n$ arbitrary ASCII bytes, where n is a non-negative integer. The following rules apply to NULL bytes appearing in ASCII COMMENT, if n is non-zero:

1. There must be at least one NULL byte in the ASCII COMMENT;
2. If a NULL byte appears in the ASCII COMMENT, all the following bytes in the ASCII COMMENT must also be NULL bytes.

The reason for these two rules is so that if the ASCII comment has nonzero length then it may always be treated as a C NULL-terminated string, with no information "hidden" after the NULL byte.

The DATA region consists of N `COMPLEX8` quantities. Each `COMPLEX8` is made of a 4-byte IEEE-754 float real part, followed by a 4-byte IEEE-754 float imaginary part. The packing and normalization of this data comply with the LSC specifications for frequency-domain data [1].

The HEADER fields are further described below:

- `version` **(Version 2)**: Shall be 2.0. **(Version 3)**: Shall be 3.0.

  Note SFTs produced before this specification will have this field set to 1.0. Note that future versions of this specification will have `version`=4.0, 5.0, etc. This field will always be an integer value that can be exactly represented as an IEEE754 double. If this field is not an exact integer in the range 1 to 1000000, then software reading this data should assume that it is byte-swapped and take appropriate measures to reverse the byte ordering. If byte swapping does NOT cause the version number to be an exact integer between 1 and 1000000, then the SFT does not comply with these specifications.

- `gps_sec`: Integer part of the GPS time in seconds of the first sample used to make this SFT.

- `gps_nsec`: GPS nanoseconds of the first sample used to make this SFT. This must lie in the range from 0 to $10^9 - 1$ inclusive.

- `tbase`: The time length in seconds of the data set used to make this SFT. This must be greater than zero.

  Note If the sample interval is dt, and the number of time-domain samples is S, then `tbase` $= Sdt$. Note that if the data is produced with heterodyning, `tbase` still refers to the total time length of the data set. Note that the frequency spacing in between samples (df, as defined in [1]) is 1.0/`tbase`. If df can not be exactly represented as an IEEE-754 double, then the closest IEEE-754 double to 1.0/`tbase` will be the closest IEEE-754 double to df.

- `first_frequency_index`: This is the subscript of the first complex FFT value that appears in DATA. Its allowed range is 0 to (Nyquist frequency) × `tbase`/2 $= S/2$ inclusive. Note that if S is odd, then in this document $S/2$ shall mean the integer part of $S/2$.

- `nsamples`: The number of complex samples N in DATA. Its allowed range is 1 to $S/2 + 1$ inclusive.

- `crc64`: The 64-bit CRC checksum of the $48 + 8n + 8N$ bytes that make up the SFT, with the 8 bytes labeled `crc64` set to zero. The CRC checksum will be evaluated using the polynomial

  D800000000000000 (base-16)
  1101100000000000000000000000000000000000000000000000000000000000 (base-2).

  The primitive polynomial is $x^{64} + x^4 + x^3 + x + 1$. The CRC will be initialized to all ones (∼0ULL).

- `detector`: Two characters of the form Xn characterizing the detector, following the naming convention for the channel-name prefix in the Frame format [2]. X is a single capital letter describing the site; n is the detector number. Currently allowed names are listed in Appendix C of [2]; in addition the names Xn, Yn, and Zn are reserved for implementation-specific use.

- `padding` **(Version 2)**: These two bytes will be set to zero. They are here so that all multi-byte quantities are byte-aligned with respect to the header. This may permit certain efficiencies and library usages on certain platforms/architectures.

- `windowspec` **(Version 3)**: Specifies the window function used in generating the SFT. It is decomposed into two integers `windowspecA` $\in [0, 12] \subset \mathbb{Z}$ and `windowspecB` $\in [0, 5000] \subset \mathbb{Z}$ as follows:

$$\texttt{windowspec} = \texttt{windowspecA} \times 5001 + \texttt{windowspecB}\,, \tag{1}$$

and conversely (provided `windowspec` > 0)

$$\texttt{windowspecA} = \lfloor \texttt{windowspec}/5001 \rfloor\,, \tag{2}$$

$$\texttt{windowspecB} = \texttt{windowspec} - \texttt{windowspecA} \times 5001\,. \tag{3}$$

For zero-parameter windows, `windowspecA` $= 0$ and `windowspecB` specifies the window type; for one-parameter windows, `windowspecA` specifies the window type, and `windowspecB` sets the window parameter in increments of $1/5000 = 2 \times 10^{-4}$:

$$\beta = \texttt{windowspecB}/5000 \in [0, 1] \in \mathbf{R}\,. \tag{4}$$

Allowed values for `windowspec` are listed in the following table:

Summary of `windowspec` layout.

Window	Short Name	`windowspecA`	`windowspecB`	`windowspec`
		Zero-parameter windows.		
unknown	`UNKN`	0	0	0
Rectangular	`RECT`	0	1	1
Hann	`HANN`	0	2	2
unused	n/a	0	3–5000	3–5000
		One-parameter windows.		
Tukey	`TKEY`	1	0–5000	5001–10001
unused	n/a	2	0–5000	10002–15002
unused	n/a	3	0–5000	15003–20003
unused	n/a	4	0–5000	20004–25004
unused	n/a	5	0–5000	25005–30005
unused	n/a	6	0–5000	30006–35006
unused	n/a	7	0–5000	35007–40007
unused	n/a	8	0–5000	40008–45008
unused	n/a	9	0–5000	45009–50009
unused	n/a	10	0–5000	50010–55010
unused	n/a	11	0–5000	55011–60011
unused	n/a	12	0–5000	60012–65012

Note that the range of `windowspec` fits within the 0–65536 range of an `UINT2`.

The "unknown" window where `windowspec` $= 0$ is backward-compatible with the Version 2 requirement that `padding` is set to zero (see above); this is consistent given that Version 2 does not formally record the window used to generate the SFT which is therefore unknown.

The reference implementations for each named *Window* are the LALSuite [3] functions `XLALCreate`*Window*`REAL8Window()`. The 4-character *Short Name*s are used by the SFT file naming convention defined in Section 2.1.

- `comment_length`: The number of bytes $8n$, with n a non-negative integer, that appear in ASCII COMMENT. Note that if `comment_length` is zero then the SFT contains no comment.

The complex quantities contained in DATA REGION are defined by the following equations, complying with [1].

The data set (with the native fundamental sample interval dt) is denoted by x_i for $i = 0, \ldots, S-1$. The x_i are all real. Let

$$n_k = \sum_{j=0}^{S-1} x_j \exp(-2\pi i j k / S) \tag{5}$$

be the values of the DFT with LSC sign conventions. The values in DATA REGION are

$$\texttt{data}_k = dt \times n_k \tag{6}$$

for $k =$ `first_frequency_index` to $k =$ `first_frequency_index` $+$ `nsamples` $- 1$. Note that the interesting range of k is from 0 to $S/2$ inclusive.

The allowed range of `first_frequency_index` is from 0 to $S/2$ inclusive.

The allowed range of `nsamples` $= N$ is 1 to $S/2 + 1 -$ `first_frequency_index` inclusive.

Note Here we assume that the window function is rectangular, i.e. each sample is weighted by a window function whose value is 1. If the data IS windowed then the normalization conventions of [1] still apply.

Note that if a data stream is band-limited (for instance by filtering) and then decimated or down-sampled, the values stored in DATA REGION for a given set of frequency bins will be unchanged compared to those computed with the original data set. This is true even though the sample interval dt' of the new down-sampled data set is larger than the original native sample time. In fact, except for the DC ($k = 0$) and Nyquist ($k = S/2 + 1$) frequency bins, the power spectral density P_k may be written as:

$$P_k = \frac{2}{\texttt{tbase}} |\texttt{data}_k|^2 \,. \tag{7}$$

1.2 Example SFTs

1.2.1 Example SFT 1

Consider the case where the fundamental time-domain data set consists of 16 samples, taken at a sample rate of 16 Hz. All 16 samples are $x_0 = \cdots = x_{15} = 1$ which gives $n_0 = 16, n_1 = \cdots = n_8 = 0$. Since $dt = 1/16$, we find:

$$\begin{array}{llll} \texttt{data}_0 = 1 + 0\mathrm{i} & \texttt{data}_1 = 0 & \texttt{data}_2 = 0 & \texttt{data}_3 = 0 \\ \texttt{data}_4 = 0 & \texttt{data}_5 = 0 & \texttt{data}_6 = 0 & \texttt{data}_7 = 0 \\ \texttt{data}_8 = 0\,. & & & \end{array} \tag{8}$$

If we store only $\texttt{nsamples} = 5$ frequency bins in the SFT, then DATA REGION will contain the 40 bytes corresponding to identical values for $\texttt{data}_k$:

$$\begin{array}{llll} \texttt{data}_0 = 1 + 0\mathrm{i} & \texttt{data}_1 = 0 & \texttt{data}_2 = 0 & \texttt{data}_3 = 0 \\ \texttt{data}_4 = 0\,. & & & \end{array} \tag{9}$$

These values could be obtained by considering a subset of the original SFT. Alternatively they could be obtained by low-pass filtering the original time series, and down-sampling it, and using the previous definitions. For example if the down-sampled time series had 8 samples $x_0 = \cdots = x_7 = 1$ with a sample time of $dt' = 1/8$, then $n_0 = 8$, and $n_1 = \cdots = n_4 = 0$. This gives the same values as above.

1.2.2 Example SFT 2

Consider a sinusoidal function at 2 Hz, $x(t) = 1 \times \cos(2\pi \times 2t)$. Using again 16 samples taken at a sample rate of 16 Hz,

$$\begin{array}{llll} x_{00} = 1.000000 & x_{01} = 0.707107 & x_{02} = 0.000000 & x_{03} = -0.707107 \\ x_{04} = -1.000000 & x_{05} = -0.707107 & x_{06} = -0.000000 & x_{07} = 0.707107 \\ x_{08} = 1.000000 & x_{09} = 0.707107 & x_{10} = 0.000000 & x_{11} = -0.707107 \\ x_{12} = -1.000000 & x_{13} = -0.707107 & x_{14} = -0.000000 & x_{15} = 0.707107\,, \end{array} \tag{10}$$

giving

$$n_0 = n_1 = 0 + 0\mathrm{i}\,, \quad n_2 = 8 + 0\mathrm{i}\,, \quad n_3 = \cdots = n_8 = 0 + 0\mathrm{i}\,, \tag{11}$$

and

$$\texttt{data}_0 = \texttt{data}_1 = 0 + 0\mathrm{i}\,, \quad \texttt{data}_2 = 0.5 + 0\mathrm{i}\,, \quad \texttt{data}_3 = \cdots = \texttt{data}_8 = 0 + 0\mathrm{i}\,. \tag{12}$$

If we down-sample the original data stream by a factor of two we get:

$$\begin{array}{llll} x_{00} = 1.000000 & x_{01} = 0.000000 & x_{02} = -1.000000 & x_{03} = -0.000000 \\ x_{04} = 1.000000 & x_{05} = 0.000000 & x_{06} = -1.000000 & x_{07} = -0.000000\,, \end{array} \tag{13}$$

giving

$$n_0 = n_1 = 0 + 0\mathrm{i}\,, \quad n_2 = 4 + 0\mathrm{i}\,, \quad n_3 = \cdots = n_4 = 0 + 0\mathrm{i}\,, \tag{14}$$

and

$$\mathtt{data}_0 = \mathtt{data}_1 = 0 + 0\mathrm{i}\,, \quad \mathtt{data}_2 = 0.5 + 0\mathrm{i}\,, \quad \mathtt{data}_3 = \cdots = \mathtt{data}_4 = 0 + 0\mathrm{i}\,. \tag{15}$$

2 SFT File and Directory Naming Conventions

2.1 File Naming Convention

SFT file names are to follow the conventions of [4] for class 2 Frame files, except with an extension of `.sft` rather than `.gwf`:

$\langle$*SFT-file-name*$\rangle$ ::= $\langle S\rangle$ '`-`' $\langle D\rangle$ '`-`' $\langle G\rangle$ '`-`' $\langle T\rangle$ '`.sft`'

where

- $\langle S\rangle$ is the source of the data; an uppercase single letter designation of the site. See Appendix C of [2] for the list of allowed site characters; in addition the sites `X`, `Y`, and `Z` are reserved for implementation-specific use.
- $\langle D\rangle$ is a description; any string consisting of alphanumeric characters plus underscore ('`_`'), and plus ('`+`'). In particular, the characters dot ('`.`'), dash ('`-`'), and space ('` `') are prohibited, as is any description consisting of a single uppercase letter, which is reserved for use by class 1 raw Frame files.

  Note This represents a *backward-incompatible* change to $\langle D\rangle$, relative to version 1 of this document (T040164-v1). Previously, $\langle D\rangle$ could also include a hash ('`#`') character. The motivation for this change is that hash is also commonly used as a comment character in text files, and allowing SFT file names to contain hashes could confuse a script parsing a text file containing both SFT file names and comments.

- $\langle G\rangle$ is the GPS time at the beginning of the first SFT in the file. This is either a 9-digit or 10-digit number, and is an integer number of seconds.

  If the beginning of the data is not aligned with an exact GPS second, then the file name should contain the exact GPS second just before the beginning of the data.

- $\langle T\rangle$ is the total time interval covered by the file, in seconds.

  If only 1 SFT is in the file, then $\langle T\rangle$ is `tbase` in the HEADER data.

  For multiple SFTs, if the data is aligned with exact GPS seconds, then $\langle T\rangle$ is simply the number of seconds between the beginning of the first SFT and the end of the last SFT.

  If the data is not aligned with exact GPS seconds, then $\langle T\rangle$ should be calculated from the exact GPS second just before the start of the first SFT to the exact GPS second just after the end of the last SFT. Data gaps (i.e. non-contiguous SFTs within a file) are permitted, though the SFTs in the file must be time ordered.

SFT file names will follow these additional rules for the description field $\langle D\rangle$:

$\langle D \rangle$ ::= $\langle NUM \rangle$ '_' $\langle IFO \rangle$ '_' $\langle TBASE \rangle$ '`SFT`' { $\langle PRIV \rangle$ | $\langle PUB \rangle$ } [$\langle NARROWBAND \rangle$]

where

- $\langle NUM \rangle$ is the number of SFTs in the file.
- $\langle IFO \rangle$ is a two character abbreviation of the interferometer data used to generated the SFT, matching the `detector` field in the SFT HEADER. This field must always begin with an uppercase letter.
- $\langle TBASE \rangle$ is the time-base in seconds of the SFT(s) in the file, matching the `tbase` field in the SFT HEADER.

  Note Even though the information required by $\langle IFO \rangle$ and $\langle TBASE \rangle$ is redundant with the SFT HEADER, many scripts rely on these fields, plus the GPS interval and source letter, to find files. Including this information in the file name will ensure that SFTs files are uniquely identified by these scripts.

- $\langle PRIV \rangle$ denotes a *private* SFT; one generated for private use or development, but not intended to be published and distributed across LSC clusters. $\langle PRIV \rangle$ follows the following syntax:

  $\langle PRIV \rangle$ ::= ['_' $\langle MISC \rangle$]

  where $\langle MISC \rangle$ is an optional field that contains any pertinent information about the SFT. $\langle MISC \rangle$ may *only* contain alphanumeric characters.

  Note This represents a *backward-incompatible* change to $\langle MISC \rangle$, relative to version 1 of this document (T040164-v1). Previously, $\langle MISC \rangle$ could also include underscore ('_'), plus ('`+`'), and hash ('`#`') characters. The motivation for removing '_' as a valid character for $\langle MISC \rangle$ is to simplify the parsing of $\langle D \rangle$, which can now be parsed into a finite number of sub-fields separated by '_'. The motivation for removing '`+`' as a valid character for $\langle MISC \rangle$ is to robustly distinguish $\langle MISC \rangle$ from $\langle PUB \rangle$ (see below), since the latter uses '`+`' to parse itself into sub-sub-fields. The motivation for removing '`#`' as a valid character for $\langle MISC \rangle$ is the same as for $\langle D \rangle$ (see above).

- $\langle PUB \rangle$ denotes a *public* SFT; one generated to be published and distributed across LSC clusters. Most commonly these will be generated from observing run data, but could also include e.g. data for a mock data challenge. Public SFTs *must* have globally unique names. $\langle PUB \rangle$ follows the following syntax:

  $\langle PUB \rangle$ ::= '`_O`' $\langle OBS \rangle$ $\langle KIND \rangle$ '`+R`' $\langle REVISION \rangle$ '`+C`' $\langle CHANNEL \rangle$ '`+W`' $\langle WINDOW \rangle$

  where

– ⟨*OBS*⟩ is an integer denoting the observing run data the SFTs are generated from, or (in the case of mock data challenge data) the observing run on which the data is most closely based. Use a value of 100 or greater if there is no obvious mapping to an existing observing run, e.g. for development work, when simulating data from potential future detectors, etc.

– ⟨*KIND*⟩ is either: '`RUN`' for production SFTs of $h(t)$ channels; '`AUX`' for SFTs of non-$h(t)$ channels; '`SIM`' for mock data challenge or other simulated data; or '`DEV`' for development/testing purposes.

– ⟨*REVISION*⟩ is an integer denoting the revision number of the SFT production. The revision number starts at 1, and should be incremented once the SFTs have been widely distributed across different clusters, advertised within the Continuous Wave Group as being ready for use, etc. For example, if mistakes are found in the initial SFT production run after they have been published, regenerated SFTs should have a revision number of at least 2.

– ⟨*CHANNEL*⟩ is the full channel name of the data used to produce the SFTs. The detector-specific prefix (i.e. before the colon) should be removed, and any non-alphanumeric characters should be stripped out.

Note It is presumed that names of data channels are globally unique (modulo removal of non-alphanumeric characters) and therefore satisfy the requirement that public SFTs must have globally unique names. For mock data challenges, the channel name of the mock data stream should be constructed to also be globally unique, including e.g. any parameter or versioning information relevant to the mock data challenge itself.

– ⟨*WINDOW*⟩ denotes the window function used in generating the SFTs. It is given by

⟨*WINDOW*⟩ ::= ⟨*WINDOW-NAME*⟩ [⟨*WINDOW-PARAM*⟩]

where ⟨*WINDOW-NAME*⟩ is given by the 4-character *Name* list in the "Summary of `windowspec` layout" table in Section 1.1. If the window *Name* requires a parameter, ⟨*WINDOW-PARAM*⟩ is the value of `windowspecB` written as a decimal integer without leading zeros; otherwise ⟨*WINDOW-PARAM*⟩ is omitted.

Examples:

Window	⟨*WINDOW-NAME*⟩	β	⟨*WINDOW-PARAM*⟩	⟨*WINDOW*⟩
unknown	`UNKN`	n/a	n/a	`UNKN`
Rectangular	`RECT`	n/a	n/a	`RECT`
Tukey	`TKEY`	0.001	5	`TKEY5`

Version 2 SFTs should always be named with ⟨*WINDOW*⟩ ::= `UNKN`.

- ⟨*NARROWBAND*⟩ denotes a *narrow-band* SFT, where the full frequency range of the data is split across multiple files. Let k = `first_frequency_index` denote the index of the first frequency bin in a narrow-band SFT, n = `nsamples` the number of bins in a narrow-band SFT, and T = `tbase` the SFT time base. The syntax for ⟨*NARROWBAND*⟩ is then given by:

$\langle NARROWBAND \rangle$ ::= '`_NBF`' $\langle FIRSTBIN \rangle$ '`W`' $\langle BINWIDTH \rangle$

$\langle FIRSTBIN \rangle$ ::= $\langle FIRSTBIN\text{-}FREQ \rangle$ '`Hz`' $\langle FIRSTBIN\text{-}REM \rangle$

$\langle BINWIDTH \rangle$ ::= $\langle BINWIDTH\text{-}FREQ \rangle$ '`Hz`' $\langle BINWIDTH\text{-}REM \rangle$

where

$$\langle FIRSTBIN\text{-}FREQ \rangle = \lfloor k/T \rfloor \,, \qquad \langle FIRSTBIN\text{-}REM \rangle = k - \lfloor k/T \rfloor T \,, \tag{16}$$
$$\langle BINWIDTH\text{-}FREQ \rangle = \lfloor n/T \rfloor \,, \qquad \langle BINWIDTH\text{-}REM \rangle = n - \lfloor n/T \rfloor T \,, \tag{17}$$

and conversely

$$k = \langle FIRSTBIN\text{-}FREQ \rangle T + \langle FIRSTBIN\text{-}REM \rangle \,, \tag{18}$$
$$n = \langle BINWIDTH\text{-}FREQ \rangle T + \langle BINWIDTH\text{-}REM \rangle \,. \tag{19}$$

Note The syntax of $\langle NARROWBAND \rangle$ is intended as a compromise between providing complete information and being human-readable. By parsing $\langle TBASE \rangle$ and $\langle NARROWBAND \rangle$ a script can uniquely determine k and n, and hence determine the exact frequency band covered by a narrow-band SFT. On the other hand, a human can inspect the '`FIRSTBIN-FREQ`' and '`BINWIDTH-FREQ`' fields in the SFT file name, and thereby determine the general bandwidth covered by a narrow-band SFT without further computation.

The following is a complete grammar for the $\langle D \rangle$ field:

$\langle D \rangle$ ::= $\langle D1 \rangle$ '_' $\langle D2 \rangle$ '_' $\langle D3 \rangle$ ['_' $\langle D4 \rangle$] ['_' $\langle D5 \rangle$]

$\langle D1 \rangle$ ::= $\langle NUM \rangle$

$\langle NUM \rangle$ ::= $\langle$*strictly-positive-integer*$\rangle$

$\langle D2 \rangle$ ::= $\langle IFO \rangle$

$\langle IFO \rangle$ ::= $\langle$*uppercase-letter*$\rangle$ $\langle$*digit*$\rangle$

$\langle D3 \rangle$ ::= $\langle TBASE \rangle$ '`SFT`'

$\langle TBASE \rangle$ ::= $\langle$*strictly-positive-integer*$\rangle$

$\langle D4 \rangle$ ::= $\langle MISC \rangle$ | $\langle D41 \rangle$ '`+`' $\langle D42 \rangle$ '`+`' $\langle D43 \rangle$ '`+`' $\langle D44 \rangle$

$\langle MISC \rangle$ ::= $\langle$*alphanumeric-string*$\rangle$

$\langle D41 \rangle$ ::= '`O`' $\langle OBS \rangle$ $\langle KIND \rangle$

$\langle OBS \rangle$ ::= $\langle$*strictly-positive-integer*$\rangle$

⟨*KIND*⟩ ::= ‘`RUN`’ | ‘`SIM`’ | ‘`DEV`’

⟨*D41*⟩ ::= ⟨*REVISION*⟩

⟨*REVISION*⟩ ::= ⟨*strictly-positive-integer*⟩

⟨*D43*⟩ ::= ⟨*CHANNEL*⟩

⟨*CHANNEL*⟩ ::= ⟨*alphanumeric-string*⟩

⟨*D44*⟩ ::= ⟨*WINDOW*⟩

⟨*WINDOW*⟩ ::= ⟨*WINDOW-NAME*⟩ [⟨*WINDOW-PARAM*⟩]

⟨*WINDOW-NAME*⟩ ::= 4 * ⟨*uppercase-letter*⟩

⟨*WINDOW-PARAM*⟩ ::= ⟨*positive-integer*⟩

⟨*D5*⟩ ::= ‘`NBF`’ ⟨*FIRSTBIN*⟩ ‘`W`’ ⟨*BINWIDTH*⟩

⟨*FIRSTBIN*⟩ ::= ⟨*FIRSTBIN-FREQ*⟩ ‘`Hz`’ ⟨*FIRSTBIN-REM*⟩

⟨*FIRSTBIN-FREQ*⟩ ::= ⟨*positive-integer*⟩

⟨*FIRSTBIN-REM*⟩ ::= ⟨*positive-integer*⟩

⟨*BINWIDTH*⟩ ::= ⟨*BINWIDTH-FREQ*⟩ ‘`Hz`’ ⟨*BINWIDTH-REM*⟩

⟨*BINWIDTH-FREQ*⟩ ::= ⟨*positive-integer*⟩

⟨*BINWIDTH-REM*⟩ ::= ⟨*positive-integer*⟩

2.2 Directory Naming Convention

Private SFTs may be organized into directories in any way that is convenient for their users.

Public SFTs are to be organized into directories as follows, depending on whether they are *narrow-band* SFTs, or not (i.e. a *broad-band SFT*):

⟨*broadband-SFT-file-path*⟩ ::= ⟨*SFT-common-dir-part*⟩ ‘`_BROADBAND/`’ ⟨*SFT-broadband-subdir*⟩ ‘`/`’ ⟨*SFT-file-name*⟩

⟨*narrowband-SFT-file-path*⟩ ::= ⟨*SFT-common-dir-part*⟩ ‘`_NARROWBAND/`’ ⟨*SFT-narrowband-subdir*⟩ ‘`/`’ ⟨*SFT-file-name*⟩

⟨*SFT-common-dir-part*⟩ ::= ⟨*D2*⟩ ‘`_`’ ⟨*D3*⟩ ‘`_`’ ⟨*D4*⟩

$\langle \textit{SFT-broadband-subdir} \rangle$ ::= ‘`GPS`’ $\langle \textit{G-MILLION} \rangle$ ‘`M`’

$\langle \textit{SFT-narrowband-subdir} \rangle$ ::= $\langle \textit{D5} \rangle$

where most elements are taken from the grammars in Section 2.1, with the addition of

$$\langle \textit{G-MILLION} \rangle = \lfloor \langle G \rangle / 1000000 \rfloor \,. \tag{20}$$

2.3 Example SFT Files Names

`H-1_H2_1800SFT-735627918-1800.sft` A 30-minute private H2 SFT. (The SFT file name is 34 characters in length.)
`L-1_L1_1800SFT_S2-733467931-1800.sft` A 30-minute private L1 SFT with the description “S2”. (The SFT file name is 37 characters in length.)
`H-1887_H1_1800SFT-733467931-4622400.sft` A private file with 1887 30-minute H1 SFTs with gaps in time. (The SFT file name is 40 characters in length.)
`G-1_G1_60SFT_S3hot-732465218-60.sft` A 60-second private GEO SFT with the description “S3hot”. (The SFT file name is 36 characters in length.)

`H-1_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5-1257800000-1800.sft`
`H-1_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5-1257801800-1800.sft`
`H-1_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5-1257803600-1800.sft`
`H-1_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5-1257805400-1800.sft`
`H-1_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5-1257807200-1800.sft`
5 public broad-band H1 SFTs from the O2 observing run; revision 1 of the public SFTs were created from the `H1:DCH-CLEAN_STRAIN_C02` channel, Tukey-windowed with $\beta = 0.001$. (Each SFT file name is 70 characters in length.)

`H-5_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5_NBF0010Hz0W0008Hz0-1257800000-9000.sft`
`H-5_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5_NBF0018Hz0W0008Hz0-1257800000-9000.sft`
`H-5_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5_NBF0026Hz0W0008Hz0-1257800000-9000.sft`
`H-5_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5_NBF0034Hz0W0008Hz0-1257800000-9000.sft`
`H-5_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5_NBF0042Hz0W0008Hz0-1257800000-9000.sft`
`H-5_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5_NBF0050Hz0W0008Hz0-1257800000-9000.sft`
`H-5_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5_NBF0058Hz0W0008Hz0-1257800000-9000.sft`
`H-5_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5_NBF0066Hz0W0008Hz0-1257800000-9000.sft`
`H-5_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5_NBF0074Hz0W0008Hz0-1257800000-9000.sft`
`H-5_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5_NBF0082Hz0W0008Hz0-1257800000-9000.sft`
`H-5_H1_1800SFT_O2RUN+V1+CDCHCLEANSTRAINC02+WTKEY5_NBF0090Hz0W0005Hz900-1257800000-9000.sft`
The equivalent narrow-band SFTs over the frequency range 10–95.5 Hz, each SFT containing 8 Hz of data except the last SFT which contains 5.5 Hz of data. (Each SFT file name is 89 characters in length.)

3 References

[1] S. Anderson, W. Anderson, K. Blackburn, P. B. D. Brown, P. Charlton, J. Creighton, T. Creighton, S. Finn, J. Romano, D. Sigg, J. Whelan, A. Wiseman, and J. Zweizig. Conventions for data and software products of the LIGO and the LSC. Technical Report T010095-x0, LIGO, 2001. `https://dcc.ligo.org/LIGO-T010095-x0/public`.

[2] LIGO and Virgo. Specification of a Common Data Frame Format for Interferometric Gravitational Wave Detectors. Technical Report T970130-v3, LIGO, 2022. `https://dcc.ligo.org/LIGO-T970130-v3/public`.

[3] LIGO Scientific Collaboration. LIGO Algorithm Library - LALSuite. Free software (GPL), 2018. `https://doi.org/10.7935/GT1W-FZ16`.

[4] P. Shawhan, S. Anderson, K. Blackburn, P. Ehrens, A. Ivanov, A. Lazzarini, B. Mours, D. Sigg, and J. Zweizig. Naming Convention for Frame Files which are to be Processed by LDAS. Technical Report T010150-x0, LIGO, 2001. `https://dcc.ligo.org/LIGO-T010150-x0/public`.

A Changelog

November 11, 2022 Version 3 of specification: record window applied to SFTs in header, expanded file and directory naming conventions for published SFTs.

June 27, 2005 SFT data should be finite() [not IEEE +/- Inf or NAN].

November 18, 2004 All times in SFT file names should be in seconds; simplify SFTtype description.

September 22, 2004 Added new section: SFT File Naming Convention.

September 17, 2004 After discussion with the pulsar search group, added detector type to SFT header.

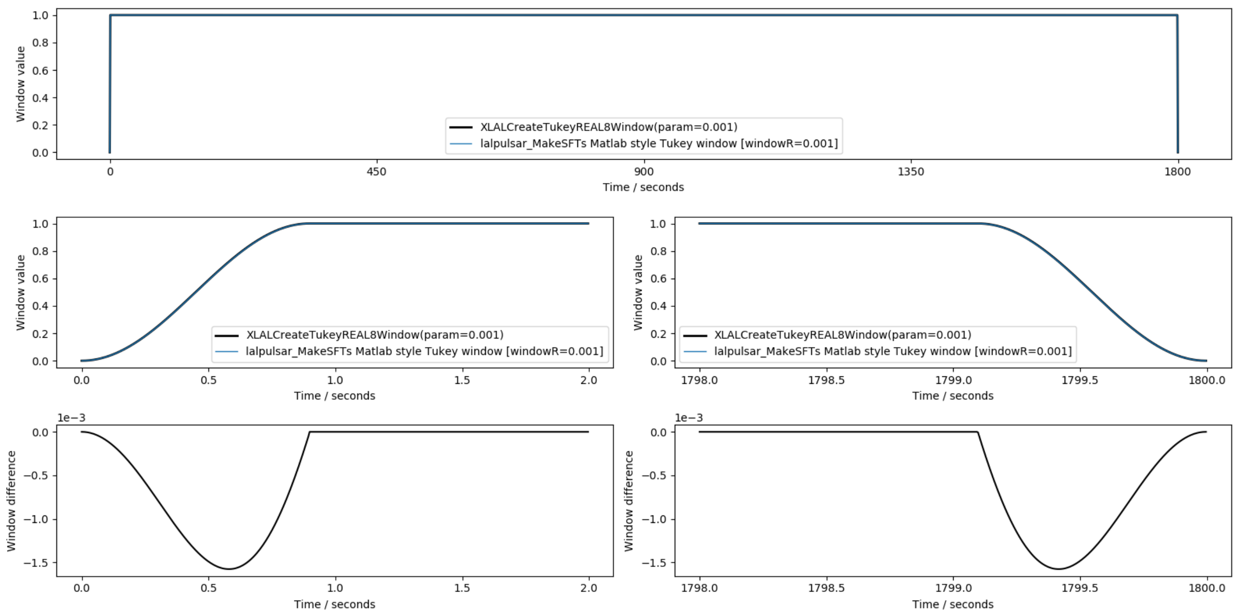
September 14, 2004 After discussions with the pulsar search group, updated spec to require that all SFT blocks share some common header info.

September 9, 2004 Release 2.3 of reference library. Spec updated as per comments in Section B.2.

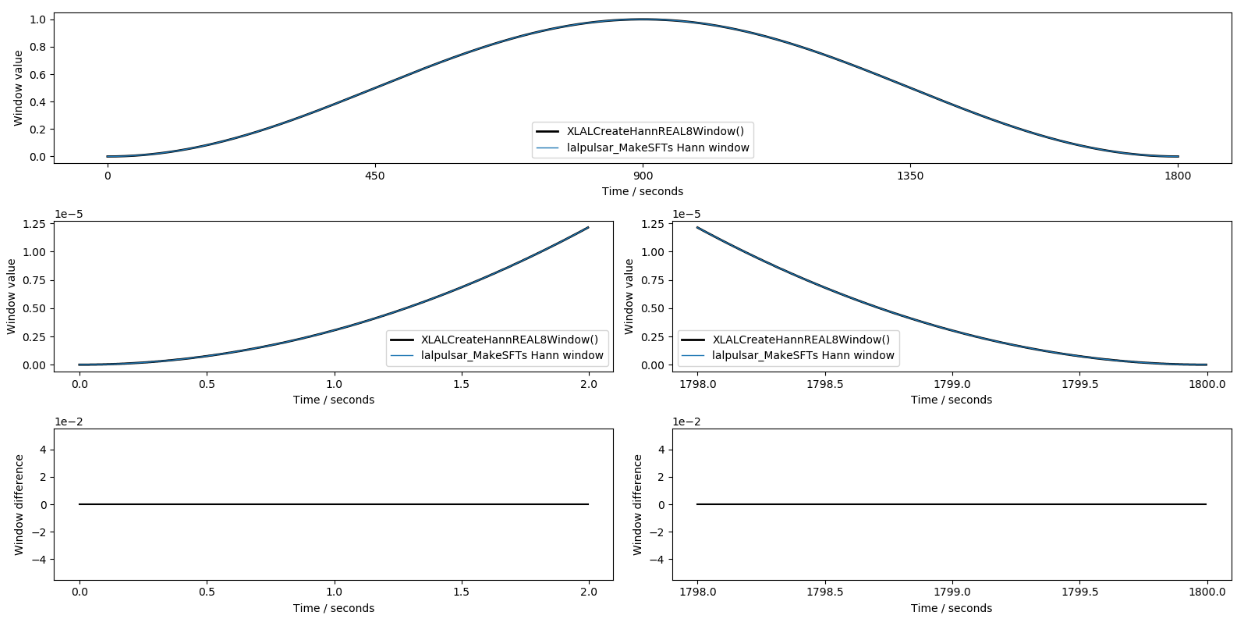
August 17, 2004 First public release of version 2.2 of reference library.

B Notes

B.1 Plots of SFT Window Reference Implementations



Plots of Tukey window reference implementation `XLALCreateTukeyREAL8Window()`, compared to original implementation in `lalpulsar_MakeSFTs`.



Plots of Hann window reference implementation `XLALCreateHannREAL8Window()`, compared to original implementation in `lalpulsar_MakeSFTs`.

B.2 2004 SCCB Report on SFT Specification

```
> From: Jolien Creighton <jolien@gravity.phys.uwm.edu>
> Cc: Stuart Anderson <sba@srl.caltech.edu>,
>     John Zweizig <zweizig_j@ligo.caltech.edu>
> Subject: SCCB report on SFT specification
> Date: Tue, 07 Sep 2004 17:44:46 -0500
>
> Here's the SCCB report on the current draft of the version 2 SFT
> specification.  We would encourage the developers to contact us if they
> have any questions.
>
> Cheers
> Jolien
> (for the SCCB)



> The SCCB requires modifications to the SFT version 2 specification
> listed in items 1-6:
>
>1. units of tbase should be defined to be seconds

Done.

>2. df should be explicitly stated to be 1/tbase

Done.  But note that in many common cases (eg, tbase=60 seconds), df
can NOT be exactly represented by an IEEE-754 double.  For example, in
this case:
1.0/60.0 =
0.001666666..... (base 10) =
2^-6 x 1.000100010001000... (base 2)
which can not be represented exactly as an
IEEE-754 double.  In this case, "approximating" 1/tbase using IEEE-754
doubles yields the 52-binary-point approximation:
2^-6 x 1.0001000100010001000100010001000100010001000100010001 (base 2)
which is (exactly)
0.0166666666666666664353702032030923874117434024810791015625 (base 10)

The fact that df (for many cases of interest) can not be exactly
represented as an IEEE-754 double is the primary reason why the SFT
format uses tbase as fundamental.

>3. padding field should be used to stored what is called S
```

Unfortunately this can not be done. The reason is that if an SFT is 'narrowbanded' this can be done in two ways. One way is to use a downsampled data set (with a smaller value of S). The other way is to simply extract a smaller set of frequency bins from an SFT sampled at a higher sample rate. These would yield identical SFT data files but with DIFFERENT values of S.

>4. reading software (or validation program) needs to check that
>   the comment contains a NUL character followed by NUL characters

Good point. We've modified the reference library and SFTvalidate code to issue an error in the case where either the comment is not NULL terminated, or in the case where a NULL character in the comment is followed by non-NULL characters.

>5. the specification must indicate conventions for the following
>   metadata within the SFT block:
>
>- full channel name if the data comes from interferometer data
>- window name (along with parameters if the name is not sufficient)
> of the window functions

The spec has been modified so that when this information is required, it is contained in the comment block.

>6. there needs to be a convention for SFT filenames

Agreed. Suggestions?

>Observations: this committee would like to explicitly note the
>following limitations of the specification:
>
>- there is not provision for specifying how many SFTs are in
> the file ... this requires that all headers be read in order
> to determine the contents and also that there is no way to
> detect "corruption" (or errors in translation) that drop
> SFT blocks from the file

It would be convenient to include this information in the SFT file name. Then it could be easily read by software using the SFT, and could also be incorporated into the SFTvalidate code.

>- there is no ability to record SFTs of complex or heterodyned
> data sets

```
We were aware of this limitation.  It was discussed at length and is a
deliberate choice.

>- future modifications to this spec will require evidence
>  that infrastructure changes to accommodate a new SFT format
>  will be significantly less than those require to convert to
>  using the frame spec

Noted.

After discussion the pulgroup has added some additional requirements
to the spec, namely that

The SFT blocks in a given SFT file are required to have:
(1) identical version numbers
(2) monotonically increasing GPS times
(3) identical values of tbase
(4) identical values of first_frequency_index
(5) identical values of nsamples
(6) come from the same instrument and instrument channel


Cheers,
Bruce Allen for the Pulsar Search Group.
```