SDP Memo 40: PSRFITS Overview for NIP

Document Number ..... SDP Memo 40
Document Type ..... MEMO
Revision C
Author ..... R. J. Lyon, L. Levin, B. W. Stappers
Release Date 2018-02-5
Document Classification ..... Unrestricted
Status ..... Draft

Lead Author	Designation	Affiliation
R. J. Lyon	SDP.PIP.NIP Member	University of Manchester
Signature & Date:		

SDP Memo Disclaimer

The SDP memos are designed to allow the quick recording of investigations and research done by members of the SDP. They are also designed to raise questions about parts of the SDP design or SDP process. The contents of a memo may be the opinion of the author, not the whole of the SDP.

Table of Contents

List of Figures

List of Tables

Summary

This document briefly summarises the PSRFITS data format, providing a high-level description relevant to Non-Imaging Processing (NIP) work. It includes an estimation of the overhead incurred when using the PSRFITS format. Calculations indicate that PSRFITS introduces a 5% file size overhead *at most*, mostly due to the manner in which the format stores metadata.

1 Introduction

PSRFITS is a standard data format used in radio astronomy. It extends the Flexible Image Transport System (FITS) format which has existed for some time (Hanisch et. al., 2001; Pence et. al., 2010). The FITS format is complicated, yet the relation to PSRFITS is relatively simple. PSRFITS only adds optional components to the standard FITS file, without altering the format in any significant way. The PSRFITS standard will be briefly described here, and the overheads incurred when utilising the standard studied. However the underlying FITS format will not be explained in detail. Interested readers can study the latest version[1] of the FITS standard on-line (FITS Working Group, 2017).

2 Background

FITS is an open standard maintained by an international working group. Whilst it was primarily designed to improve the handling and transmission of astronomical image data, it can accommodate various forms of data product with ease. Due to this inherent flexibility, the FITS format was customised to make it suitable for storing pulsar data[2] (Hotan et. al., 2004). The extended format, PSRFITS, has been adopted in radio astronomy circles. Many tools take advantage of the underlying FITS structure to simplify the handling and interoperability of data.

PSRFITS is a standard that can be used to transmit Non-Imaging Processing (NIP) data products between the Central Signal Processor (CSP), and the Science Data Processor (SDP); as disucssed the SDP-CSP Interface Control Document (ICD)[3]. There are three pipelines that will likely make use of the format. This includes,

- the pulsar search sub-element (PSS), which we call *SDP.PSS*. This filters candidates produced by the CSP, with the aim of isolating pulsar signals. The SDP.PSS pipeline processes the Optimised Candidate and Data List (OCLD) data product.
- the transient search sub-element (TSS), which we call *SDP.TSS*. This also filters candidates produced by the CSP, but with the aim of isolating fast transient signals. The SDP.TSS pipeline processes the Single-pulse Optimised Candidate and Data List (SPOCLD) data product.
- the pulsar timing sub-element (PST), which we call *SDP.PST*. This processes the data produced by the CSP, with the aim of timing the regular pulses emitted by many pulsars. This involves determining if there is a difference in pulse arrival time over time, and recording this information for analysis. The SDP.PST pipeline processes the Pulsar Timing Data (PTD) product.

The PSRFITS format was chosen as it is a de-facto standard for pulsar data, and given the wide variety of compatible tools available. Many (if not all) of these tools are open source. This includes, for example, DSPSR (van Straten & Bailes, 2011), PSRCHIVE (van Straten et. al., 2012), PRESTO (Ransom, 2011) and PSRSALSA (Weltevrede, 2017). Both PRESTO and PSRSALSA achieve compatibility via easy to use python interfaces.

[1] See version 4.0.

[2] Including n-dimensional data cubes, and simple time series.

[3] PSRFITS is not necessarily the format that will/should be used. Improved standards could emerge in the time between design and implementation, e.g. HDFITS Price et. al. (2015).

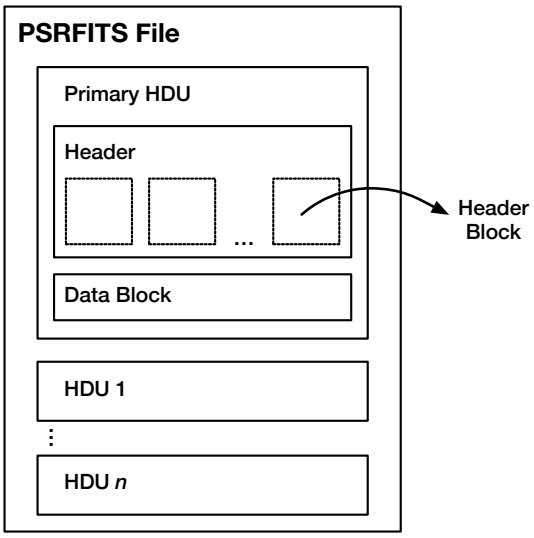

Figure 1: Overview of the structure of a PSRFITS file.

3 Format Overview

The FITS data model is closely linked to the FITS file format (Price et. al., 2015). Therefore the format has some quirks due to the way the data is physically stored. A FITS file has the following basic components:

- Primary header and data unit (HDU).
- Conforming extensions (optional).
- Other special records (optional, restricted).

PSRFITS files adhere to this format, and include a number of pulsar specific conforming extensions. These extensions are all that differentiate a PSRFITS file from a standard FITS file. It is necessary to better understand these components, before we can estimate the overhead incurred by using PSRFITS for NIP work. Readers interested in in gaining a deeper understanding of the format, should read the PyFITS user guide[4].

3.1 Primary Header and Data Unit

Each PSRFITS file must contain a primary HDU. This is comprised of two logical parts, a *header* and a *data block* as shown in Figure 1. The header stores metadata that enables the data within the HDU to be correctly decoded. Whilst the data block contains information stored as a simple byte stream (big-endian ordering).

3.1.1 Header

According to the FITS specification (FITS Working Group, 2017), the header is composed of one or more *header blocks*. Each block contains 36 records. Each record[5] is an 80 character entry, described via a restricted set of ASCII text characters (1 byte per character). The

[4] See `http://stsdas.stsci.edu/stsci_python_sphinxdocs_2.13/pyfits/users_guide/users_guide.html`.

[5] Called a 'card' in the FITS specification.

```
SIMPLE = T / conforms to FITS standard
BITPIX = 8 / bits per data pixel
NAXIS = 2 / number of data axes
NAXIS1 = 256 / length of data axis 1
NAXIS2 = 256 / length of data axis 2
```

Figure 2: Example PSRFITS file HDU header block.

syntax for an individual record consists of a single keyword-value pair: *keyword* $=$ *value*. The keyword is up to 8 characters long[6]. The keyword must be followed by a whitespace character, then an equals sign, and then another whitespace character. This is expected before any value is given. The symbol '/' can be used to begin a comment, but only after the value is given.

Together the pairs (including comments) cannot exceed 80 characters. If more than 80 characters are needed, the specification describes a way to break metadata across multiple lines, but we do not consider this here. An extract from a valid header block is shown in Figure 2.

Each header block is exactly 2,880 bytes in size[7]. In practice there can be many header blocks, so long as the final block terminates with the 'END' keyword. This keyword marks the logical end of the header. Records without a value, or not using all 80 characters, are filled with ASCII spaces. Thus *every* header block is always exactly 2,880 bytes in size. This size is the result of design choices relating to how the data is serialised. Note there are currently 5 mandatory records per file. This means that at a minimum, each PSRFITS file will contain 1 header with 31 spaces for any required additional information.

3.1.2 Data Block

This contains the data, as described by the header. This is stored as a simple byte stream that can be decoded using the information within the header. The data block is simple, and contains no other information which can incur a file size/transfer overhead.

3.2 Conforming Extensions & Special Records

Conforming extensions are very similar to the primary HDU, in that they contain a header and a data block. The extensions specify the type of data they contain more formally than the primary HDU. There are three primary extension types:

- 'IMAGE' extensions.
- 'TABLE' ASCII table extensions.
- 'BINTABLE' binary table extensions.

Extensions effectively define custom data types. The 'IMAGE' extension can be used to store multi-dimensional arrays, the 'TABLE' extension provides a way to create a simple ASCII text table, and the 'BINTABLE' extension allows raw data to be stored efficiently in large tables. Special records are used to store information that poses an unstructured format.

[6] Can be lengthened in extensions, but these are not considered.
[7] This is given by $36 * 80 = 2,880$ bytes.

The 'BINTABLE' extension is utilised by PSRFITS to store some forms of pulsar data, e.g. the sub-integrations. There is a metadata overhead associated with using binary tables. The overhead is variable, depending on what information is required to interpret the data (e.g. offsets, weights, scalings etc). In general the number of metadata items equals $rows \times attributes$, where rows is the number of rows in the table, and attributes are the number of pieces of separate information required to decode/interpret the row contents.

3.3 PSRFITS Extensions

PSRFITS consists of a primary HDU, followed by a series of extension HDUs (header plus data). In practice the primary HDU describes for example, the telescope used to obtain the data, the observation parameters etc. All other extension HDUs, are formatted as binary tables. According to the PSRFITS documentation[8], these contain information such as the pulsar ephemeris, calibration data and pulsar data formatted as a series of sub-integrations. There are 14 PSRFITS extensions, and these are listed in Table 1. Each extension has its own appropriate header[9]. All extensions do not have to be used, and their data contents can be set to empty, if only metadata need be stored.

4 Estimating the Overhead

4.1 Method

Two types of analysis are used to determine the overhead incurred by using the PSRFITS format. The first involves estimating the expected overhead directly from the file format specification. The second involves undertaking some real-world tests with the ASTROPY library[10], with the aim of determining if the estimates are reasonable.

4.2 Assumptions

We assume that all PSRFITS extensions are used, thus incurring the maximum metadata overhead possible (an unrealistic scenario). The cost of each header is given in Table 1, where each header character is assumed to be represented using 8-bit ASCII text (1 byte). Data compression is not considered given the costs of compression/decompression at both ends of the pulsar data processing chain. Note that transmission overheads are not considered. Finally we emphasise that it is not possible to know exactly what metadata will be needed in practice. Thus it is not possible to estimate potential overheads exactly.

5 Results

5.1 Estimate from the Specification

The maximum possible amount of metadata a PSRFITS file can store, can be calculated from Table 1. The theoretical size of all metadata is $40,000$ bytes, or 40 kB (kilobytes) given the assumptions above. To store the same information in a PSRFITS file requires $57,600$ bytes

[8] See Hotan et. al. (2004) and `http://www.atnf.csiro.au/people/pulsar/index.html?n=Main.Psrfits`.

[9] Described fully on-line here: `http://www.atnf.csiro.au/people/pulsar/index.html?n=PsrfitsDocumentation.Txt`.

[10] See `http://www.astropy.org`, version 2.0 used.

Table 1: A summary of the current PSRFITS extensions and the primary header (V5.4). Columns 1 and 2 are taken directly from the PSRFITS web-page. The 'Size (bytes)' column indicates the exact size for the records making up each extension header in bytes. The 'PSRFITS Size (bytes)' gives the size of the header when in PSRFITS format, which requires that each header block contain exactly 36 entries (2,880 bytes). If there are too few entries in a block (i.e. 35 instead of 36), the block is padded with whitespace. If there are too many records for one block, a new block must be created and padded when appropriate. Note that each header character is assumed to be represented using 8-bit ASCII text (1 byte).

HDU Name	Description	Records	Size (bytes)	PSRFITS Size (bytes)
Main header	Observer, telescope and receiver information, source name and observation date and time.	62	4,960	5,760
HISTORY	Date, program and details of data acquisition and each subsequent processing step.	70	5,600	5,760
OBSDESCR	Free-format ascii description of the observation or signal processing.	11	880	2,880
PSRPARAM	Pulsar ephemeris used to create or modify pulse profile data.	11	880	2,880
POLYCO	History of the TEMPO polyco files used to predict the apparent pulsar period.	38	3,040	5,760
T2PREDICT	The TEMPO2 predictor file used to predict the apparent pulsar period.	11	880	2,880
COHDDISP	Parameters used for coherent dedisperion of baseband data.	36	2,880	2,880
BANDPASS	Observed bandpass in each polarisation averaged over observation.	19	1,520	2,880
FLUX_CAL	System temperature and injected noise calibration data as a function of frequency across the bandpass.	35	2,800	2,880
CAL_POLN	Apparent polarisation of injected noise calibration signal as a function of frequency.	18	1,440	2,880
FEEDPAR	Parameters of feed cross-coupling as a function of frequency.	32	2,560	2,880
SPECKURT	Statistics for spectral kurtosis RFI excision.	26	2,080	2,880
SUBINT	Pulse profiles or streamed data as a function of time, frequency and polarisation.	88	7,040	8,640
DIG_STAT	Digitiser mode, attenuator settings and count statistics.	20	1,600	2,880
DIG_CNTS	Digitiser mode and count rate distribution.	23	1,840	2,880
TOTAL		500	40,000	57,600

or 57.6 kB. The larger size is due to the requirement that each header block be comprised of exactly 36 entries, totalling 2,880 bytes each. If a header has less than 36 records, it will be padded with empty whitespace entries as necessary. This increases the file size. The difference between the theoretical and actual values, yields an estimated overhead of 17,600 bytes (17.6 kB). This means the PSRFITS format adds a 36% file size overhead for metadata. This does not include metadata costs incurred via storing data in binary tables, e.g. the sub-integrations.

A 36% overhead appears expensive. However metadata forms only a small fraction of the file contents. The much larger data block portion of the file (containing the pulsar observation) incurs no appreciable overhead, as it is simply a byte stream[11]. Thus the overhead is small overall. The following scenario provides some context.

Suppose a single pulsar candidate data cube is to be stored in a PSRFITS file. The dimensions of the cube are $128 \times 128 \times 64$, where each sample is 1 byte in size. The total size of the cube is 1,048,576 bytes, which is just over 1 MB (megabyte). Suppose this data were to be combined with all the PSRFITS header extensions, adding an additional 57.6 kB to the total file size. The size of the resulting combined data product is 1,106,176 bytes. Here the 17.6 kB metadata overhead is approximately 1.6% of the total data product. This is very low. Furthermore this percentage decreases as the data block becomes larger. This will be shown in the following section.

5.2 Empirical Tests

Using the ASTROPY Python library, a series of FITS files were generated and analysed. The procedure used was as follows:

1. Generate a FITS file using ASTROPY.
2. Compute the size of the file in bytes.
3. Calculate the smallest theoretical size of the file (number of samples $\times$ bytes per sample).
4. Calculate the overhead as the difference between the observed and expected theoretical file size.
5. Calculate the overhead as a percentage.
6. Record the result.
7. Repeat as required.

Incrementally larger FITS files were created using a simple loop. On each loop iteration a 3-dimensional data cube was generated and stored in the file. The dimensions of the cube ranged from (1,1,1) up to (300,300,300), such that each unique triple (x,y,z) formed a unique independent test. Each cube sample was encoded as an 8 byte floating-point value. Thus the cube ranged in size from 8 bytes up to 216 MB. During these empirical tests only the mandatory header information was included. This was done to test the overhead incurred via

[11] Test results are described in Section 5.2 that support this view.

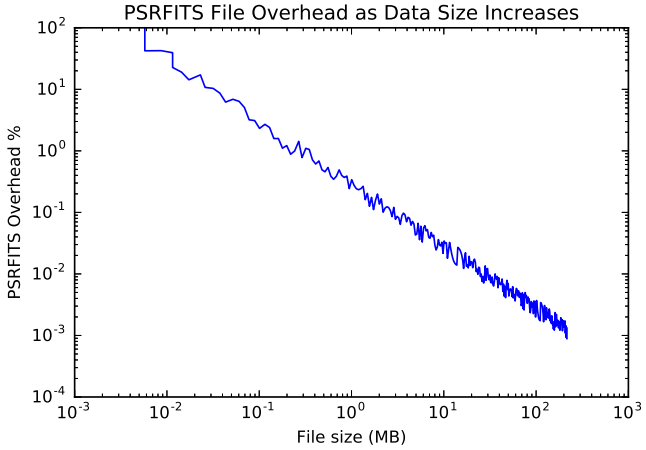

Figure 3: The overhead incurred when using the FITS format as the data unit component increases in size. Both the x and y axes are presented using a log scale for clarity.

increasing the data block size in isolation.

The results are shown in Figure 3. When the data block is very small, the size of the metadata header is comparatively large. This causes the percentage overhead to reach approximately 99%. This makes sense as the most basic header is much larger than the smallest 8 byte data cube tested. However as the data block increases in size, the situation reverses. When the block reaches 1 MB in size, the overhead drops to 0.3%. When the data cube is at its largest (216 MB), the overhead falls to 0.0008%. This suggests that the data block incurs little or no storage overhead. If this were not the case, we would expect to see the percentage overhead increase, which we do not. The result agrees with both the FITS documentation and our expectations.

6 Implications

Previously we studied NIP data rates and data product sizes (Lyon, 2017). In that work, the estimated size of an OCLD was approximately 1.08 GB (gigabaytes) per scan, per telescope beam. This estimate assumed 1,000 candidates per OCLD[12], a candidate list containing 100,000 entries, and assumed no file format overhead. Using this information we assess the impact of the overhead measured in earlier sections.

Lets assume a scenario where the PSRFITS format incurs a 53 kB overhead per candidate. This is 35.4 kB more (≈100% greater) than the estimate of Section 5.1[13]. We use this larger value to accommodate overheads which we cannot fully model or understand at this point in time. For example, the possible overhead incurred via using binary tables to store sub-integration and sub-band data. If there are 128 sub-integrations, and 3 metadata variables per sub-int each 8 bytes in size, the additional overhead is 3,072 bytes (3.072 kB). While this suggests the binary table overhead is low, we do not decrease our metadata estimates leaving room for unexpected additions and changes.

[12]With 1 data cube & 3 signal-to-noise (S/N) sheets per candidate.
[13]We use a larger value to provide a reasonable buffer.

In our scenario the total overhead per OCLD is 53 MB per beam, per scan. Adding this 53 MB cost to the OCLD (producing a 1.134 GB product) corresponds to an overhead of ≈5%. If we multiply the OCLD size by 1500 beams, the total data volume per scan is ≈1.701 TB (terabytes). If this data were to be transmitted over 100 seconds, the corresponding data rate would be ≈136 Gbit/s. The data rate without the PSRFITS overhead is ≈130 Gbit/s. Thus the difference between the two is a marginal ≈6 GBit/s. This would not impact the SDP architecture at scale. If in practice more transfer time is available, the data rate will be lowered further. Based on these numbers it is best to assume a maximum overhead of 5% for the PSRFITS format. The maximum cost is incurred if all extensions are used, where each binary table in the extension HDU has data. Note we only consider the SDP.PSS scenario here, as it exhibits the highest NIP data rates and volumes[14].

7 Conclusions

The cursory analysis presented here suggests the PSRFITS format introduces a small storage overhead. In practice this is less than 5%. However it is advisable to expect an overhead of 5%. This provides room for future metadata changes if required. Based on existing data models and data rate estimates (Lyon, 2017), a 5% overhead has negligible impact on the SDP, is not a source of significant architectural risk. Moving forward it is important to define and finalise the metadata that will be needed to assist SKA data processing. Only when that work is complete, can we accurately determine file format overheads. It is advisable to ask the science working groups to assist in this task, as their help will be invaluable.

8 Acknowledgements

Thanks to Willem van Straten for helping us understand the size of the 'BINTABLE' component of a PSRFITS file.

References

FITS Working Group, 2017, "Definition of the Flexible Image Transport System (FITS), version 4.0", `https://fits.gsfc.nasa.gov/standard40/fits_standard40aa.pdf`.

Hanisch R. J. et. al., 2001, "Definition of the Flexible Image Transport System (FITS)*", A & A, 376, 1, pp.359-380. doi:10.1051/0004-6361:20010923.

Hotan A. W. et. al., 2004, "PSRCHIVE and PSRFITS: An Open Approach to Radio Pulsar Data Storage and Analysis", Publications of the Astronomical Society of Australia, Vol. 21, Issue 3, pp. 302-309. doi:10.1071/AS04022.

Lyon R. J., 2017, "CSP to SDP NIP Data Rates & Data Models (version 1.1)". doi:10.5281/zenodo.836715.

Pence W. D. et. al., 2010, "Definition of the Flexible Image Transport System (FITS), version 3.0", A & A, 524. doi:10.1051/0004-6361/201015362.

[14] Compared to the SDP.TSS and SDP.PST pipelines.

Price D. C. et. al., 2015, “HDFITS: Porting the FITS data model to HDF5”, Astronomy and Computing, Vol. 12, Supplement C, pp. 212-220. doi:10.1016/j.ascom.2015.05.001.

Ransom S., 2011, “PRESTO: PulsaR Exploration and Search TOolkit”, Astrophysics Source Code Library. http://adsabs.harvard.edu/abs/2011ascl.soft07017R

van Straten W. & Bailes M., 2011, “DSPSR: Digital Signal Processing Software for Pulsar Astronomy”, PASA, 28, 1. doi:10.1071/AS10021

van Straten W. et. al., 2012, “Pulsar Data Analysis with PSRCHIVE”, Astronomical Research and Technology, vol.9, No.3, pp.237-256. doi:10.1071/AS10021

Weltevrede P., 2017, “PSRSALSA: A Suite of Algorithms for Statistical Analysis of pulsars”. http://www.jb.man.ac.uk/~wltvrede/psrsalsa.pdf