Description of the VDIF Extended Data Version 4: Multiplexed VDIF Data Validity

Walter Brisken

07 Jan 2016

1 Introduction

The VDIF[1] data format has brought a new level of sophistication and flexibility to encapsulation of VLBI baseband data. Its very high rate of adoption globally has already led to simplication in exchange of this data. A feature of VDIF that illustrates its flexibility, and that is of key importance to this proposed extension, is the ability to store fully corner-turned data (e.g., one thread with many channels), data that has not been corner-turned (e.g., multiple threads, each with containing a single channel), and partially corner-turned data (multiple threads, each with multiple channels).

There are cases, involving software that has been built around processing of fully corner-turned data, where it is advantageous to convert VDIF data that is either not corner-turned or is partially corner-turned into data that is fully corner-turned. In this document that process will be called *multiplexing*. A problem with doing this is that data validity, as indicated by the "invalid bit" of the VDIF header, becomes impossible to represent with the single "invalid" bit of the multiplexed frame. The consequence is that a choice must be made by the software performing the multiplexing: "should invalid data be processed, or should valid data be discarded?" In the case that missing data is very rare this distinction is of little concern, but there are cases (see below) where a large fraction of data, typically from one thread at a time, is lost. This document describes VDIF Extended Data Version (EDV) number 4, which provides a mechanism to handle this.

2 Use cases

There are two general classes of use cases for this VDIF EDV.

The first is intermediate on-disk formats. For various reasons it may be useful to convert the native output from some backend/recorder system, which may be a single disk file itself or multiple files that must be interleaved, to a single serialized output. In this step missing frames, either from individual threads or across all threads, may be inserted such that the output stream has no gaps in its time sequence. Such needs have arisen in the diagnostics and early use of Mark6 data (in the "gather" process). The EDV4 will play an especially important role in cases where one or more disks in the module set becomes unusable between recording and processing.

The second is completely internal use where the EDV4 VDIF variant is never written to disk. An example is DiFX software correlator. The core of this correlator was built around datastreams with fully corner-turned data. The effort involved in reengineering DiFX for native operation on arbitrary input VDIF data is expected to be quite high. The intermediate solution adopted is for the data reading process to corner-turn the data before sending to the core of DiFX. EDV4 will allow proper accounting of "data weights" on a per baseband channel basis.

[1] Defined in document http://vlbi.org/vdif/docs/VDIF_specification_Release_1.1.1.pdf.

2.1 Use case to avoid

It is highly discouraged to make use of this form of VDIF as the native product of any data source.

3 Implementation

The structure of the VDIF header with EDV4 extension is shown in Table 1. Words 4 through 7 contain the new information being described here; words 0 through 3 remain unchanged and no change in the VDIF version itself is required.

Byte #	3								2								1								0							
Bit #	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	I_1	L_1	Seconds from reference epoch$_{30}$																													
Word 1	U_2			Epoch$_6$					Data frame number in second$_{24}$																							
Word 2	Ver$_3$			$\log_2(n_{\text{chan}})_5$					Data frame length (units of 8 bytes)$_{24}$																							
Word 3	C_1	Bits/samp-1$_5$							Thread Id$_{10}$								Station Id$_{16}$															
Word 4	EDV=4$_8$								Validity mask length$_8$								U_{16}															
Word 5	Sync word = 0xACABFEED$_{32}$																															
Word 6	High bits of validity mask$_{32}$																															
Word 7	Low bits of validity mask$_{32}$																															

Table 1: A VDIF frame header when employing EDV 4. Subscripted numbers indicate the number of bits used by the field. Data in words 0 through 3 are described in the VDIF specification document. Some abbreviations are used: "I" is the invalid bit, "L" is the legacy VDIF bit, "U" indicates unused bits, "Ver" indicates the VDIF version, "C" is the complex data bit. Data in words 4 through 7 form EDV 4. Individual fields are described in the text. The last two words should be interpreted as a single 64-bit bitfield. Note that all fields use Intel byte order (little-endian).

There are only three fields of importance:

- **Validity mask length**: an integer with a value between 1 and 64 (inclusive) indicating the number of data validity bits.
- **Synchronization word**: a 32-bit sequence that can be used to ensure (or rediscover) synchronization of the VDIF data sequence. The value (0xACABFEED) and location (word 5) of this are the same as employed by EDVs 1 and 3.
- **Validity mask**: A 64-bit sequence indicating validity of data on a channel-by-channel basis. In the case that the validity mask length equals the number of channels in the thread there is a one-to-one mapping of validity mask bit to channel. The mask for the first channel (channel 0) is stored in the least significant bit of this 64-bit word (bit zero of word 7). If the bit is set (1) that means valid data is present, otherwise (0) data for this channel should not be considered valid. The sense of this mask was consciously chosen to be opposite to that if the "data invalid bit" such that unused bits (bit numbers greater than or equal to the validity mask length) are left unset (0).

3.1 EDV4 and the VDIF "data invalid" bit

The VDIF standard stipulates that EDV data is supplemental and thus it should not be required for general use. This leaves the question open to how the frame "data invalid" bit should be set in cases where some, but not all, of the channels within the frame contain valid data. This choice impacts how downstream consumers of VDIF data that do not respect EDV4 handles such situations. Since there is no correct answer to this question, and in fact this question is what led to the development of EDV4, it is suggested here that the software or human producing data in EDV4 format use their own judgement. Software performing the multiplexing should allow options so that users can select an algorithm best suited for the particular application. Such software should also report on the fraction of time an incomplete (but not empty) frame has been constructed.

3.2 The case of more than 64 channels

In the case that a single VDIF frame contains more than 64 channels the EDV4 concept is extended by having each bit in the validity mask represent the state of more than one channel. The number of channels in a single VDIF frame is always a power of two. Thus the number of channels divided by the 64 would always be an integer, n. The first n channels would respect the value of the least significant bit in the validity mask, and in general the validity of channel c (0-based) would be represented by bit $\lfloor c/n \rfloor$ (also 0-based) of the validity mask. The same judgement call described above is needed when assigning a single validity bit to multiple channels. The choice is out of the scope of this document.

3.3 Unused bits

There remains 16 bits of unused space in the EDV4 header. To allow for potential future use of these slots there is no restriction on their contents. They can be considered to be usage-specific, and no software should depend on these bits taking on any particular value.