The DVB-T2 Reference Streams

Version: 1.2 (21st January 2021)

1. Overview and scope

The present document describes the generation of reference streams used for the verification and the validation of the DVB-T2 specification. The methods were developed by the verification and validation (V&V) group of DVB TM-T2 to ensure that the specification was unambiguous and that all companies involved had the same understanding of it. The resulting streams are now being released publicly to allow other implementers to verify their implementations.

This document defines reference streams generated at different *test points* in the modulator chain and for different *test cases*. The parameters of the test cases are defined in a spreadsheet.

During the V&V process, each company generated its own reference streams at some or all of the test points for each test cases and compared them against the streams generated by the other companies. The crosschecking started from the input and gradually advanced towards the output. The released streams have all been verified by several independent implementations.

First, the modulator block diagram is introduced, together with the definition of the test points at which streams are generated. Next, the input stream generation is described, comprising the generation of input packets and the assembly of these packets into input Transport Streams. In order to give a deterministic output signal to compare, the implementation of certain non-prescriptive parts of the specification are then defined. Some test cases involve FEFs, and the content of these are defined. Next, the stream file format is defined, supporting the various data types in the modulator chain. The structure of the file format is then presented.

The present document was based on a working document of the V&V group.

2. Block diagram and test points

The block diagram of the DVB-T2 modulator is shown in **Figure 1**, together with the test points at which streams are generated. The input generator and processing, as well as the bit-interleaved coding and modulation (BICM) are *PLP specific, i.e. one per PLP*, whereas the frame builder and the OFDM generation are common. In MISO mode there are *two paths* starting from the output of the "MISO Processing" block. The two paths are identical, except for the "Cell Multiplexer" block, which is slightly different for the two transmitters.

The "Annex D Splitting" block will have no effect when common PLPs are not used and test point 0 need not be generated in this case.
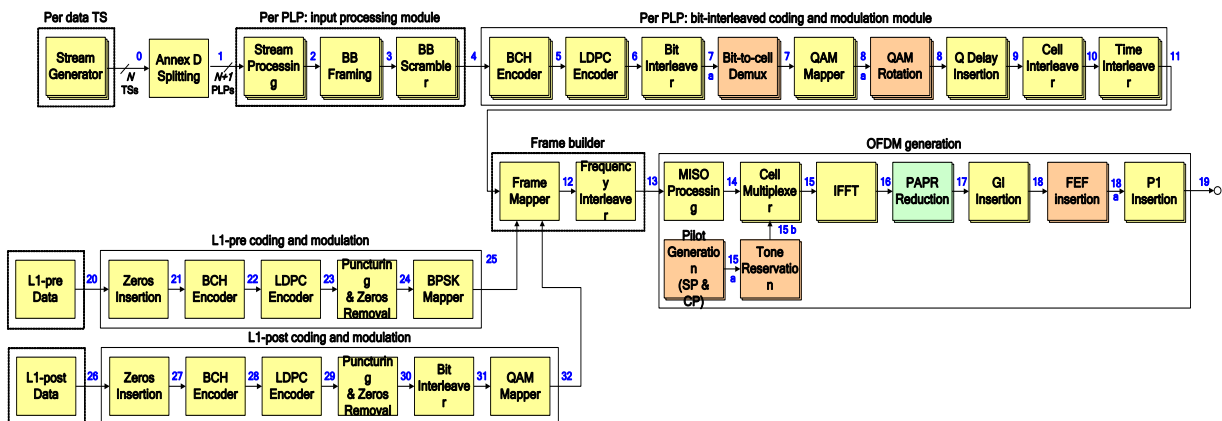

Figure 1. Block diagram of the DVB-T2 modulator

The input stream generator generates ISO/IEC 13818-compliant Transport Stream (TS) packets with a pseudo-random payload and is described in the next section. Only TS inputs are supported at this time.

The published DVB-T2 reference files were generated by the DVB Common Simulation Platform, which does not currently generate test points 2, 15a, 15b or 18.

The data formats at the test points are as follows:

- Points 0—3: byte

- Points 4—7, 20—24, 26—31: bit

- Points 8—19, 25, 32: complex floating point

How the data at these test points is written to file is explained in the section 6.

The proposed method supports one or more PLPs.

For test cases based on the simple input stream model (4.1), only the CBR (constant bit rate) scenario is verified. For each PLP, the constant data rate will be specified in terms of FEC blocks per frame. The actual bit rate in Mbps will then depend on: 1) the FEC code rate and block size and 2) the input processing.

For test cases based on the dynamic multiple-PLP model (4.2) the input bit-rate is the same for all PLPs within a group and is discussed in section 5.3.

3. Input packet generation

The Input Stream generator generates packets of the following different packet types. How the various packet types are assembled into Transport Streams is described in section 4.

3.1. Normal Packets

Normal packets have a pseudo-random payload in the data_byte field. A TS packet has a length of 188 bytes and starts with a sync_byte having the value 47_{16}. The values of the various fields are defined in Table 1 below.

TS Packet field	Length (bits)	Value
sync_byte	8	`01000111`
transport_error_indicator	1	`0`
payload_unit_start_indicator	1	`0`
transport_priority	1	`0`
PID	13	`1000000000000 + PLP_ID`
transport_scrambling_control	2	`00`
adaption_field_control	2	`01`
continuity_counter	4	Initialized to `0000` and increments for every normal TS packet with the same PID
data_byte	184×8	ITU-T O.151, length 2^{23}-1 PRBS payload

Table 1. TS Packet definition: Normal packets

Note that the top bit of the PID is set to avoid those reserved according to EN300468, clause 5.1.3.

The payload (data_byte field) shall be pseudo-random data that is produced using an external LFSR of sequence length 2^{23}-1 with the generator polynomial $x^{23}+x^{18}+1$, as specified by ITU-T recommendation O.151, clause 2.2. The implementation is shown in **Figure 2**.
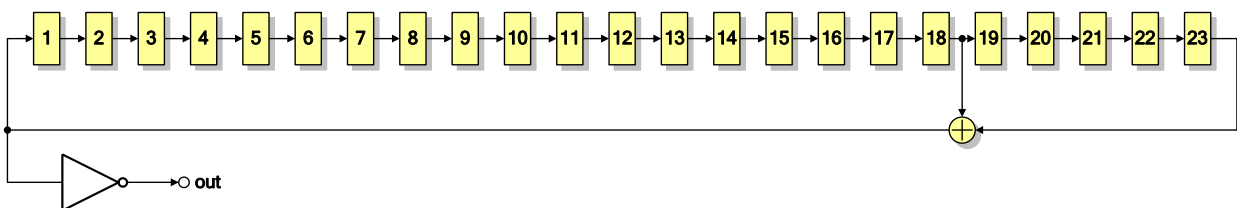

Figure 2. The 23 register LFSR

The generator is initialized once at the beginning of the simulation by loading the LFSR with a predetermined initial value. *In order to distinguish between the multiple PLPs, the LFSR will be loaded with the binary complement of the PLP_ID*, as shown in **Table 2** for PLP_ID from 0 to 3 (note that the LSB is on the left).

TM-T2_0587

PLP_ID	Initial state
0	111111111111111111111
1	011111111111111111111
2	101111111111111111111
3	001111111111111111111

Table 2. Initial states and generated binary sequences

Note that the generator is *not* initialized at the beginning of a new frame or super-frame.

Figure 3 shows the conversion from bits to bytes using the "MSB first" convention. The same convention will be used for serialization in the input processing part of the modulator.
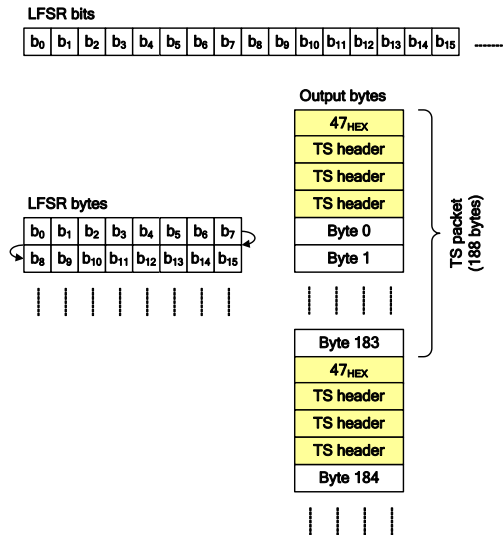

Figure 3. Conversion from bits to bytes and the generation of TS packets

For subsequent Normal Packets in a given Transport Stream, the payload consists of the subsequent bytes generated from the LFSR, i.e. the LFSR shall not be "clocked" while null packets or common packets are being inserted in the stream.

3.2. Null Packets

Null packets shall be as defined in the following table:

TS Packet field	Length (bits)	Value
sync_byte	8	01000111
transport_error_indicator	1	0
payload_unit_start_indicator	1	0
transport_priority	1	0
PID	13	1111111111111 (1fff_{16})
transport_scrambling_control	2	00
adaption_field_control	2	01
continuity_counter	4	Always 0000
data_byte	184×8	All 0

Table 3. TS Packet definition: Null packets

For Null Packets the data_byte field shall be all zeros and the continuity_count shall always be 0000.

3.3. Category-1 common packets

Category-1 common packets are generated in the same way as normal packets – see 3.1. The PLP_ID used to generate the PID value and to initialise the LFSR shall be the PLP_ID of the corresponding common PLP.

3.4. Category-2 common packets (SDT)

Category-2 common packets shall contain an Service Description Table describing the single service carried in a particular PLP. A "category-2 common packet describing PLP x" describes the service in PLP x; note that versions of the packet will be carried in all the input Transport Streams.

The following rules shall be followed when generating the Service Description Table:

- No descriptors in loops;

- Original_network_id set to 0x0000;

- Only one service in SDT loop (N=1);

- Each input TS shall be considered to contain a single service;

- The Service_ID for a given service shall be equal to 0x1200 plus the PLP_ID for the data PLP in which the corresponding service is carried;

- The Transport_stream_ID for a given TS shall be equal to 0x3400 plus the PLP_ID of the data PLP in which the corresponding TS is carried;

- SDT is in Running mode.

Table 1.   Service description section

Syntax	Length (bits)	Format	Value
service_description_section(){			
table_id	8	uimsbf	• 01000010: SDT actual; • 01000110: SDT other.
Section_syntax_indicator	1	bslbf	1
reserved_future_use	1	bslbf	1
reserved	2	bslbf	11
section_length	12	uimsbf	000000010001
transport_stream_id	16	uimsbf	See above
reserved	2	bslbf	11
version_number	5	uimsbf	00000
current_next_indicator	1	bslbf	1
section_number	8	uimsbf	00000000
last_section_number	8	uimsbf	00000000
original_network_id	16	uimsbf	0000000000000000
reserved_future_use	8	bslbf	11111111
for (i=0;i<N;i++){			N is fixed to 1.
Service_id	16	uimsbf	See above.
Reserved_future_use	6	bslbf	111111
EIT_schedule_flag	1	bslbf	0 for Category 2 streams, 1 if Category 2 and 3 are combined.
EIT_present_following_flag	1	bslbf	0 for Category 2 streams, 1 if Category 2 and 3 are combined.
Running_status	3	uimsbf	100
free_CA_mode	1	bslbf	0
descriptors_loop_length	12	uimsbf	000000000000
}			
CRC	32	rpchof	Dynamically computed
}			

- For the TS that the packet describes, the *table_id* shall be 0x42, SDT actual

- For the other TSs, *table_id* shall be 0x46, SDT other.

TM-T2_0587

See **Figure 6:** for the second co-timed packets, *transport_stream_id and service_id* fields in the SDT other table are set to 0000000000000000 as Input 2 and Input 3 share the same SDT other.

3.5. Category-3 common packets (EIT)

Category-3 common packets shall contain an Event Information Table describing the single service carried in a particular PLP. A "category-3 common packet describing PLP x" describes the service in PLP x; note that versions of the packet will be carried in all the input Transport Streams.

The following rules shall be followed when generating the Event Information Table:

- No descriptors in loops;
- *original_network_id* set to 0x0000;
- Only one event in EIT loop (N=1);
- Each input TS shall be considered to contain a single service;
- The Service_ID for a given service shall be equal to 0x1200 plus the PLP_ID for the data PLP in which the corresponding service is carried;
- The Transport_stream_ID for a given TS shall be equal to 0x3400 plus the PLP_ID of the data PLP in which the corresponding TS is carried;
- All timing information are set to:
  - *start_time*: 0x0000000000;
  - *duration*: 0x000000.
- EIT is in *running* mode.

Table 2.   Event information section

Syntax	Length (bits)	Format	Value
event_information_section(){			
table_id	8	uimsbf	0x4E: EIT actual present/following 0x50 – 0x5F: EIT actual schedule 0x4F: EIT others present/following 0x60 – 0x6F: EIT others schedule
section_syntax_indicator	1	bslbf	1
reserved_future_use	1	bslbf	1
reserved	2	bslbf	11
section_length	12	uimsbf	000000011011
service_id	16	uimsbf	See above
reserved	2	bslbf	11
version_number	5	uimsbf	00000
current_next_indicator	1	bslbf	0
section_number	8	uimsbf	00000000
last_section_number	8	uimsbf	00000000
transport_stream_id	16	uimsbf	See above
original_network_id	16	uimsbf	0000000000000000
segment_last_section_number	8	uimsbf	00000000
last_table_id	8	uimsbf	See below
for (i=0;i<N;i++){			N is fixed to 1.
Event_id	16	uimsbf	0x5600 plus PLP_ID
start_time	40	bslbf	0000000000000000000000000000000000000000
duration	24	uimsbf	000000000000000000000000

running_status	3	uimsbf	100
free_CA_mode	1	bslbf	0
descriptors_loop_length	12	uimsbf	000000000000
}			
CRC	32	rpchof	Dynamically computed
}			

- For the TS that the packet describes, the *table_id* shall be 0x4E or 0x50, EIT actual

- For the other TSs, *table_id* shall be 0x4F or 0x60, EIT other.

3.6. Mapping of sections into TS packets

The service_description_section and the event_information_section shall be mapped into their corresponding SDT and EIT TS packets in the following way, as mandated by the SI specifications for DVB (section 5.1.2 of the DVB BlueBook A038):

- The first byte after the 4-byte TS header is a 0x00 pointer_field, which says that the section starts at the very next byte. The reason why we need a pointer_field is that a new payload unit is started in each SDT and EIT packet. This also means that the corresponding payload_unit_start_indicator bit in the TS header shall be 1 for SDT and EIT packets.

- The section data and its associated CRC-32 are mapped right after the pointer_field

- If the section length is less that the payload length (184 bytes), the remaining bytes will be stuffed with 0xFF.

4. Input stream generator

The input stream generator generates an input transport stream for each data PLP by assembling packets of the types described above.

Different models are used to define the input streams for different test cases. The following input stream generation models are defined.

4.1. Non-dynamic model with no Null Packets

This model corresponds to a configuration in which there is either a single fixed-bitrate PLP or multiple independent PLPs each with a fixed bit-rate. The input Transport Stream for each PLP consists entirely of a sequence of Normal Packets generated according to section 3.1.

4.2. Dynamic multiple PLP model

This models the case of multiple PLPs with dynamically varying bit-rates.

- The input TSs for all the PLPs within a group have identical bit-rates and time-aligned packets as required by annex D of the specification.

- For normal packets, null packets are inserted in a complementary manner in each of the input TSs, i.e. one TS carries the normal packet and the others carry a null packet. This ensures constant total data rate whilst allowing variation of the individual data rates.

- Common packets may also occur and a common PLP may be used

- Null packet deletion is used, since this is essential to achieving a variable bit-rate given a fixed-rate Transport Stream

- Allocation model 2, described in the Implementation Guidelines, is used.

- ISSY is mandatory and is calculated according to section 5.2.

The input TS bit-rate shall be defined in the Parameter Sets spreadsheet but must be chosen so as to ensure that neither the total data rate of data PLPs nor the data rate of the common PLP is exceeded. Designers of modes must take into account the allocation model (see 5.3), the proportion of packets that will be carried in the common PLP, and the effect of common packets of category 2, versions of which are

carried in both the data *and* the common PLP. It is intended that a spreadsheet will be developed to make this calculation easier.

There are *N* Transport Streams corresponding to *N* data PLPs.

A set of time-aligned packets across the PLPs will be referred to as a "slot". Each slot can be of any of the following types:

- Normal packet slot: The next normal packet from the packet generator for one of the TSs, and a null packet in the other TSs

- A category-1 common packet slot: identical category-1 common packet in all the TSs

- A category-2 common packet slot: all TSs contain an SDT according to section 3.4. This will be SDT actual in the TS it describes, and SDT other in the other TSs

- A category-3 common packet slot: all TSs contain an EIT according to section 3.5. This will be EIT actual in the TS it describes, and an EIT other in the other TSs

Additional slot types could be added to test the splitting and merging rules further, including slots with tables not meeting the appropriate rules, and "null packet slots" with a null packet in all TSs.

The slot types follow a regular sequence in which one in *M* slots is a common packet slot and the remainder are normal packet slots. The normal packet slots are made up of series of "chapters"; in each chapter is made up of a whole number of repeats of a repeating unit; the repeating unit itself is made up of runs of normal packets for each TS in turn.

The pseudocode below generates a sequence of slots with the following characteristics:

- A repeating pattern is defined involving a run of normal packets for each TS in turn.

- The length of each run is determined by the test case (see below)

- This repeating pattern is repeated a number of times; this constitutes one "chapter" of the simulation

- A new repeating pattern is then established for the next chapter, and the process repeats for each chapter

- A common packet slot is inserted every *M* packets.

The following parameters will be defined:

N: Number of Transport streams (equal to the number of data PLPs)

M: Interval in packets between common PLP slots (defined in the Parameter Sets spreadsheet).

L: Number of separate "chapters" (defined in the Parameter Sets spreadsheet)

N_{EIT}:   Number of successive common packet slots carrying EIT packets for a given PLP (defined in the Parameter Sets spreadsheet)

```
PacketIndex = 0;

for chapter = 0..L-1 {

    NumReps = GetNumReps(chapter);

    for I = 0..N-1 {

        RunLength[i] = GetRunLength(I, chapter);

    }

    for rep=1..NumReps {

        for I = 0..N-1 {

            for j=1..RunLength[i]   {

            if (mod (PacketIndex, M) == 0) {

                Common Packet Slot (see below for sequence);

                PacketIndex++;

            }
```

```
                    Normal packet slot for TS I;

                    PacketIndex++;

            }

        }

}
```

A list of *L* values of NumReps and *N* lists of *L* values of RunLength[i] for each PLP are provided in the Parameter Sets spreadsheet. The values in each list correspond to the chapters of the simulation in chronological order. The functions GetNumReps(chapter) and GetNextRunLength(I, chapter) return the chapter from the relevant list for the relevant chapter.

A Normal Packet Slot for TS *I* contains a normal packet in the Transport Stream corresponding to data PLP *I*, and a null packet in all other Transport Streams.

In Common Packet slots, the slots are allocated in strict rotation in the following sequence:

- A category-1 common packet slot

- A category-2 common packet slot describing TS 0

- A category-2 common packet slot describing TS 1

- …

- A category-2 common packet slot describing TS *N-1*

- N_{EIT} category-3 common packet slots carrying EIT schedule describing TS 0

- …

- N_{EIT} category-3 common packet slots carrying EIT schedule describing TS *N-1*

-

- A category-3 common packet slot carrying EIT present/following describing TS 0

- …

- A category-3 common packet slot carrying EIT present/following describing TS *N*-1

Figure 4 shows an example.

Figure 4: Example sequence of slots and packets

TM-T2_0587

5. Non-prescriptive parts of the standard

Some parts of the DVB-T2 standard are not completely prescriptive, since they are expected to be performed once in the T2-gateway. However, for the purposes of the V&V exercise, a prescriptive method is required for them in order that all participants can generate identical files.

5.1. Normal mode and SYNCD

In normal mode, a CRC-8 is calculated across each user (transport stream) packet and appended after the associated user packet in the BBFRAME. The SYNCD in the BB header then points to the first CRC-8 present in the data field.

For the purposes of the V&V process, a 'dummy' CRC-8 (value 00_{16}) shall be inserted in the first byte of the data field of the very first BBFRAME and the SYNCD shall point to this (i.e. SYNCD=0000_{16}).

5.2. ISSY data generation

For the purpose of the V&V process, we define here how the ISSY data should be generated. (Note that this order is now mandated in Annex C of version 1.2.1 of the specification).

The ISSY fields should be transmitted in the following order:

- The first ISSY field of an Interleaving Frame (IF) contains the TTO   data (row 5 of Table C.1).

- The second ISSY field of an IF contains the BUFS data (row 3 of Table C.1).

- The third and following ISSY fields of an IF contain the ISCR data (row 1 or 2 of Table C.1).

The TTO data consists in the fields TTO_E, TTO_M and optionally TTO_L. In order to have a unique representation and the maximum possible precision, the TTO value should be encoded using the smallest possible TTO_E.

Example: TTO=1983488 gives TTO_E=14, TTO_M=121 and TTO_L=16.

The *design delay* value is needed in calculating TTO, and shall be specified in the Parameter Sets spreadsheet, in units of the fundamental time period *T*. The design delay is defined and discussed in the Implementation Guidelines[1].

The BUFS data consists in a field giving the unit and a value. The value of BUFS shall be specified in the Parameter Sets spreadsheet. In order to have a unique representation and the maximum possible precision, the BUFS data should be encoded using the smallest possible unit.

Example: BUFS=2079152 (the maximum) gives BUFS_unit=8Kbits and BUFS_value=256

The value in the ISCR field should be generated from the real ISCR value with the following formula:

$$ISCR\_field = floor(ISCR + 0.5)$$

The ISCR value begins at zero.

5.3. Allocation model

This concerns the way the packets are allocated to BBFrames and these BBFrames allocated to Interleaving Frames (and hence T2-frames).

For the non-dynamic model (4.1), the number of BBFrames (and hence FEC blocks) per Interleaving Frame is constant, and there is no common PLP. In this case the bit-rate for each PLP is constant and is considered to equal the input TS bit rate exactly. The input packets are inserted directly into the BBFrames and BBFrame padding is never used (except for in-band 9ignalling).

For the dynamic multiple PLP model (4.2), allocation is more complicated. The input bit-rates of all PLPs need to be the same to allow the splitting-and-merging method (spec annex D) to be used. However, since

---

[1] (TR 102 831 / DVB Blue Book A133) clause 8.8.2 onwards

the bit-rates will vary dynamically after null packet deletion, it is necessary to use a more sophisticated allocation model. For simplicity in the V&V group, allocation model 2 of the implementation guidelines shall be used. This also has the benefit of testing the use of BBFrame padding.

In the following, the symbols are as defined in clause 6.3.2 of the Implementation Guidelines.

The allocation is then perfomed as follows:

- The "collection window" for the first Interleaving Frame begins with the first bit of the input TSs.

- All collection windows are exactly $T_{cw} = P_I \times I_{JUMP} \times (T_F + T_{FEF} / I_{FEF})$ long.

- The collection windows follow contiguously one after the other with no overlap.

- All the bits arriving in the collection window for a particular Interleaving Frame are allocated to that Interleaving Frame

- The number of TS bits in T_{IF} will generally not be an integer. A bit will be considered to have arrived during the collection window if the *end* of the bit period falls within the window.

- The *beginning* of the first bit period of the input TSs will be aligned to the beginning of the first collection window.

- The packets are mapped into BBFrames, after Null Packet Deletion and Mode Adaptation, as appropriate

- Each PLP is allocated the minimum number of BBFrames needed to contain the data received in the collection window.

- The last BBFrame of the Interleaving Frame for each PLP shall be padded as necessary using the BBFrame padding mechanism.

- If the total number of BBFrames allocated to the common and/or data PLPs is less than the maximum, dummy cells shall be used to make up the T2-frame capacity; i.e. there shall be no blocks allocated to a PLP which contain only BBFrame padding.

- The last complete user packet of each Interleaving Frame for each PLP shall always be transmitted, i.e. it shall not be deleted even if it is a null packet and NPD is being used. (This is required to conform to Annex C of version 1.2.1 of the specification).

5.4. Scheduling

PLPs of the same type (common, type 1 or type 2) shall be mapped to the T2-frame in order of PLP index, i.e in order of their appearance in the spreadsheet.

Version 1.1.1 of the specification mandated that the PLPs be mapped contiguously to the cell addresses, i.e. no dummy cells were allowed between the slices of common or type 1 PLPs, nor between the subslices of type 2 PLPs. Test cases for which the specification version is specified in the spreadsheet as 1.1.1 shall therefore perform scheduling in this way.

Version 1.2.1 and later allowed dummy cells to be inserted in between slices and/or subslices. Some test cases require this to be performed in order to meet the requirements of the Receiver Buffer Model. Such test cases have the entry "Pseudo Fixed Frame Structure" set to "Yes" in the spreadsheet; this is explained in the next section. Where this entry is "No", PLPs shall be mapped to the cell addresses as for version 1.1.1.

5.4.1. Pseudo-fixed frame structure

"Pseudo-fixed frame structure" indicates that the common, type 1 and type 2 regions start at the same address in each T2-frame, and the SUBSLICE_INTERVAL remains the same in each T2-frame, regardless of the variation in the dynamic parameters. (The phrase "pseudo-fixed" is used since within the PLPs the frame structure is not fixed, it is just some of the PLP starts that are fixed)

The line "pseudo-fixed frame structure" controls whether or not fixed cell parameters are used.

Three additional parameters are provided for use with Pseudo Fixed Frame Structure, which are at the bottom of the spreadsheet. These describe the maximum number of cells of each type (common, type 1 and type2) that will be allowed by the frame structure, and when working in Pseudo fixed mode, the starts of each type will be fixed according to these.

The type 1 PLPs shall always start "max_common_cells_per_t2_frame" after the start of the common PLPs, the type 2 PLPs shall always start "max_type1_cells_per_t2_frame" after the type1s and the aux streams (if

any) shall always start "max_type2_cells_per_t2_frame" after the type2s. Also, the sub-slice interval is fixed to max_type2_cells_per_t2_frame/num_subslices. In all cases, dummy cells will fill in any gaps.

5.5. Annex-D splitting model

The splitting process shall transfer all packets into the common PLP which the rules allow to be transferred. (Note that this is now mandatory following the revision of Annex D in version 1.2.1 of the specification).

5.6. Compensating delay

Where a compensating delay is required (as described in clause 5.1.4 of the specification), the delays shall be calculated as described in clause 8.9 of the Implementation Guidelines.

The compensating delay shall not produce any output until the delay time has elapsed, i.e. no bits or packets will be output and therefore no BBFrames will be generated. Since this would not be compliant with certain requirements of the DVB-T2 specification (e.g. the minimum of 3 BBFRAMEs per Interleaving Frame), the final output, TP19, shall start from the beginning of the first superframe for which all the input data is valid. This shall be indicated by the first "# frame" line having a frame number greater than 1 (see the description of the file format in section 7).

6. FEF Parts

There are a number of possible FEF parts that are generated as part of the V&V process (TP18a).

6.1. Null FEF

A FEF part where no signal shall be generated (0 sample values) apart from the appropriate P1 symbol.

6.2. PRBS FEF

Data based on the generation of the P1 symbol shall be used to fill the FEF part. Figure 5 shows the generation of the data, whilst Figure 6 shows how this data is mapped to the FEF part.
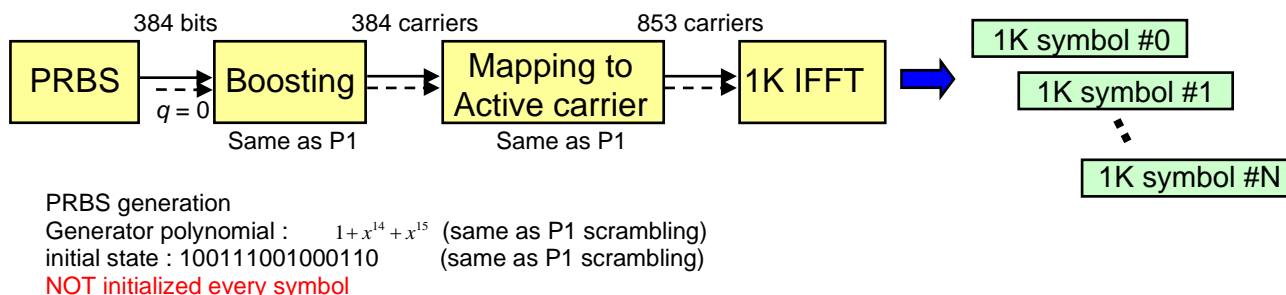

PRBS generation
Generator polynomial : $1 + x^{14} + x^{15}$ (same as P1 scrambling)
initial state : 100111001000110 (same as P1 scrambling)
NOT initialized every symbol

Figure 5. Generation of symbol data for insertion into the FEF part

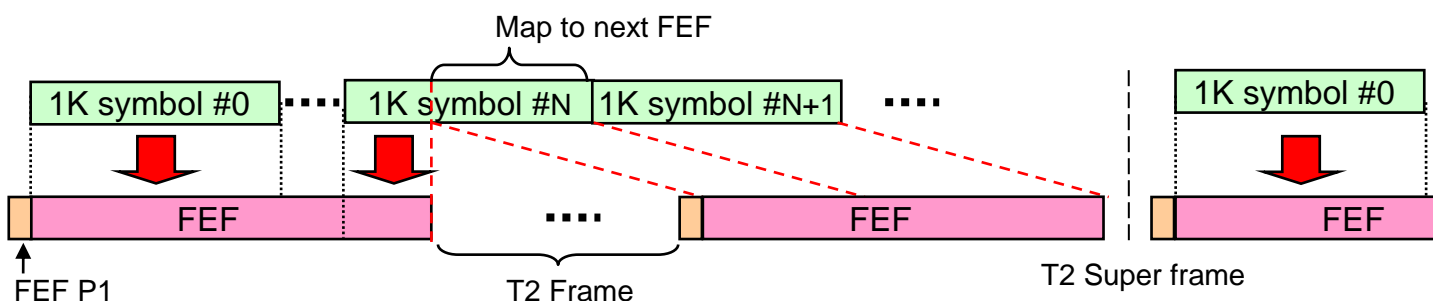The PRBS shall be reset to the initial state every superframe.

Figure 6. Mapping of symbols to FEF part

TM-T2_0587

6.3. Tx Signature

The values to be used will be specified in the Parameter Sets spreadsheet.

7. Stream file format

The file format supports the four data types encountered in the modulator: byte, bit, integer, and complex floating point. Moreover, the data hierarchy is defined to consist of blocks and frames.

Bit data is written using a *one-digit binary format*. The following `printf` format string is used: "`%1d`". Each line contains 64 characters after which a new line character is inserted.

Byte data is written using a *two-digit HEX format*. The following `printf` format string is used: "`%02X`". Each line contains 64 characters after which a new line character is inserted.

Integer data is written using a *two-digit HEX format*, regardless of the constellation size. The following `printf` format string is used: "`%02X`". The range of values depends on the constellation size:

BPSK:	`00 ... 01` (for L1 signalling)
QPSK:	`00 ... 03`
16-QAM:	`00 ... 0F`
64-QAM:	`00 ... 3F`
256-QAM:	`00 ... FF`

For the above data types, the HEX format will use uppercase letters only. Moreover, each data block will be followed by a line break. Each line contains 64 characters after which a new line character is inserted.

Complex floating-point data is written using the *exponential notation with lowercase e*, with at least 6 digits for the fractional part. In order to ensure that the field width is the same for both positive and negative numbers, which improves the readability of the file, the former will be preceded by a "+" sign. For example, the resulting string for $\pi/10$ will be: `+3.141593e-001`. The number of digits in the fractional part is 6 and the number of digits in the exponent is 3 (using zero-padding).

The real and the imaginary part will be separated by a space character and each complex pair will be followed by a newline character. The following `printf` format string is used: "`%+e %+e\n`".

Note: The published streams were generated by the CSP on a Linux platform and so the new line consists of a LF character only.

A **block** of data, such as a FEC block or an OFDM symbol will begin with the following line:

```
# block n of N
```

The block number `n` is 1 for the first block of each frame, and `N` gives the total number of blocks in the frame.

Note that not all the blocks in a frame will have the same size. One such example is the output of the frame mapper, since the P2 symbols contain fewer cells than the data symbols.

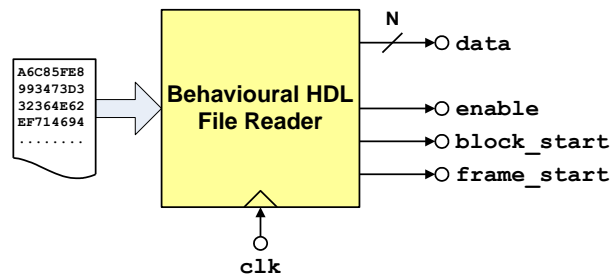A **frame** will begin with the following line:

```
# frame n
```

The frame number `n` shall start from 1 for the first frame of the first superframe and increase for each subsequent frame. However, where compensating delays are used, the first frame number occurring in the file will not necessarily be 1 (see section 5.6). As in the case of `block`, anything that comes after the frame number `n` will be treated as a comment.

A FEF part is considered as a separate `frame` in the files. In order to preserve the sequence of T2-frames, the frame number `n` indicated for the FEF part is halfway between the two T2-frames between which it is inserted, e.g. the FEF part between T2-frames 2 and 3 will begin:

```
# frame 2.5 (FEF)
```

Any line that starts with the character `%` is a comment and should be ignored by a file reader. Typically, the file header will contain a number of comment lines.

The stream files can be generated and read by both SW and behavioural HDL modules. For instance, an HDL file reader would read the file and generate the corresponding data at the output, accompanied by possibly three control signals: `enable`, `block_start`, and `frame_start`, as shown in **Figure 7**. The former is generated internally by the HDL module with a timing that depends on the specifics of the actual design. The latter two, however, can be directly derived from the `#block` and `#frame` keywords.

Figure 7. Reading a stream file by a behavioural HDL file reader module

A hypothetical example of a HEX stream file is shown below:

```
% Comment line 1
% Comment line 2
# frame 1 of 2
# block 1 of 4
A6C85FE8993473D332364E62EF7146949B31456814C0EB7937F88379A65F0EEC
# block 2 of 4
145A2B1A7CBEE5B30B87E99DC923E072FB870A0181DDB2A8FAC76D1226DC0686
# block 3 of 4
AA460F215375FE0B468F6F4BA8BA22F208EA357F2DA6367193694494DFB591EE
# block 4 of 4
D49065226197BBA0158A6DBF8519C61424787E8FAF3A4AB143AD5CA096E0767C
# frame 2 of 2
# block 1 of 3
5C702B72593B3E4C3419A86AAAAF3B31977AC5A6DD4998D451FA7A8A8B8F3111
# block 2 of 3
675CACEF32B188D76AB852940C37B5B6...
```

8. Downloading the stream files

The DVB-T2 streams are available from the DVB website at:

https://dvb.org/specifications/verification-validation/dvb-t2-reference-streams/

As well as the streams themselves, the page also contains links to the present document together with the spreadsheet `T2StreamsParameterSets<vv>.xlsx` that defines the configurations. `<vv>` is the version number of the corresponding V&V group working document on which it is based. Each configuration is identified by a unique string `<VVReference>`, which is defined in row 8 of the Parameter Sets spreadsheet and is always in the format:

VV<nnn>-<mnemonic>

where: `<nnn>` is a unique number (zero-padded to 3 digits) incremented for each new configuration and `<mnemonic>` is a short code that describes the significant parameter of the configuration such as FFT size or coderate.

There is one archive file in zip format for each configuration. Each zip file has the name:

<VVReference>_<company>.zip

where `<VVReference>` is the identifier for the configuration as defined in the spreadsheet (see above). The `<company>` part indicates the implementation that generated the files and for the published streams is always `CSP`.

8.1. Structure of archive files

Each archive file shall contain a subdirectory for each test point, having the name `TestPointXX`, where `XX` is the *number of the test point* in zero-padded two-digit decimal format. Each test-point directory will contain stream files from different companies. The files will obey the following naming convention:

<center><code><VVReference>_TP<xx>_<company>.txt</code></center>

where `VVReference` is the configuration name and `<xx>` is the number of the test point, as defined in section 2 and zero-padded to two digits. As above, `<company>` indicates the implementation that generated the files and for the published streams is always `CSP`.

For test points that have an 'a' or 'b' suffix, these shall be included in the relevant base test point directory, i.e. `<VVReference>_TP08a_<company>.txt` shall reside in the directory `TestPoint08`.

For example, the archive named:

<center><code><VVReference>_CSP.zip</code></center>

and shall contain test point files with the name and directory structure:

<center><code>TestPointXX/<VVReference>_TPXX/<VVReference>_TPXX_CSP.txt</code></center>

8.1.1. Multiple PLPs

When more than one PLP is used there will be multiple files associated with certain test points (those marked 'per-PLP' in Figure 1). In this case, the filename for those test points (and those test points only) is extended to include a number that references an individual PLP as follows:

<center><code><VVReference>_TP<xx>PLP<y>_Company.txt</code></center>

where `VVReference` is the VV Reference, `xx` is the number of the test point and `y` is the reference number for the PLP.

Note that the PLP reference number does not necessarily map to the same PLP_ID. The PLP_ID for a given PLP reference number is specified as an entry in the Excel file '`T2StreamsParameterSets<vv>.xlsx`' in the `docs` directory.

For Test point 0, the signals are TSs prior to the Annex-D splitting process. In this case the PLP number in the filename will refer to the *data* PLP associated with that TS. Note that there will therefore be no file for the *common* PLP (if used) at this test point.

8.1.2. MISO

Where test cases specify MISO processing there will be two sets of OFDM generation (test points 14-19 inclusive) that correspond to each of the two MISO transmitter groups. The filename shall be formatted as follows:

<center><code>VVReference_TP<xx>Tx<z>_<company>.txt</code></center>

where `VVReference` is the configuration name, `<xx>` is the number of the test point and `<z>` is the number of the MISO transmitter group, either '1' or '2'.

9. Version History

Version	Date	Description
0.1	15/09/2010	First version created from original V&V group working document. PRBS initialisation patterns corrected to 23 bits
1.0	23/09/2010	Version published on FTP site (no changes from 0.1)
1.1	11/08/2011	Added details of "pseudo-fixed frame structure")
1.2	21/01/2021	Updated to give new location of stream files