

## CHAPTER 7

### Packet Telemetry Downlink

<b>Chapter 7. Packet Telemetry Downlink.....</b>	<b>7-1</b>
7.1 Packet Telemetry .....	7-1
7.2 Packet Telemetry Data Packet Structure.....	7-2
7.2.1 PTDP Header .....	7-3
7.2.2 PTDP Payload.....	7-4
7.2.3 PT Package Fragmentation .....	7-10
7.3 Packet Telemetry Data Frame Structure .....	7-11
7.3.1 PTFR Header .....	7-11
7.3.2 PTFR Payload .....	7-13
7.4 Asynchronous PTDP Multiplexing.....	7-13
7.4.1 Standard PTDP Encapsulation.....	7-14
7.4.2 Low-Latency PTDP Encapsulation.....	7-14
7.5 PT Data Frame Transport.....	7-15
<b>Appendix 7-A. Extended Binary Golay Code .....</b>	<b>A-1</b>
A.1. Introduction.....	A-1
A.2. Encoding Golay Code.....	A-2
A.3. Decoding Golay Code.....	A-2
A.4. Decoding the Golay Code (8,1,3) .....	A-4
<b>Appendix 7-B. Citations .....</b>	<b>B-1</b>

### List of Figures

Figure 7-1. Packet Telemetry Overview.....	7-2
Figure 7-2. PTDP Structure .....	7-3
Figure 7-3. PTDP Header Structure (Unencoded).....	7-3
Figure 7-4. PT Fill Packet.....	7-4
Figure 7-5. PT Test Counter Packet (Unencoded).....	7-4
Figure 7-6. Chapter 10 General Packet Structure .....	7-5
Figure 7-7. PT Chapter 10 Packet Structure .....	7-5
Figure 7-8. PT Chapter 10 Header Protected Fields (Unencoded).....	7-6
Figure 7-9. PT Chapter 10 Header Unprotected Fields .....	7-6
Figure 7-10. PT Raw Ethernet MAC Frame Packet .....	7-8
Figure 7-11. PT IPv4 Packet.....	7-8
Figure 7-12. PT IPv6 Packet.....	7-8
Figure 7-13. Chapter 24 TmNSMessage Structure.....	7-9
Figure 7-14. PT TmNSMessage Packet Structure .....	7-9
Figure 7-15. PT TmNSMessage Header Protected Fields (Unencoded).....	7-10

Figure 7-16.	PT TmNSMessage Header Unprotected Fields .....	7-10
Figure 7-17.	PT Packet Fragmentation and Reassembly.....	7-11
Figure 7-18.	Packet Telemetry Data Frame (PTFR) Overview.....	7-11
Figure 7-19.	PT Data Frame Structure .....	7-11
Figure 7-20.	PTFR Header Unprotected Fields.....	7-12
Figure 7-21.	PTFR Header Protected Fields (Unencoded).....	7-12
Figure 7-22.	PTFR Payload Structure .....	7-13
Figure 7-23.	Start of PTDPs Overview.....	7-14
Figure 7-24.	Standard PTDP Encapsulation Overview .....	7-14
Figure 7-25.	PTDP Encapsulation with LLPs Overview.....	7-15
Figure 7-26.	Splitting a PTFR into Two Segments .....	7-16
Figure 7-27.	PCM Minor Frame With Two PTFR Segments .....	7-16
Figure A-1.	Golay Code Encoding and Decoding.....	A-1

## Acronyms

IP	Internet Protocol
IPv4	Internet Protocol, Version 4
IPv6	Internet Protocol, Version 6
LLP	low-latency PTDP
MAC	media access control
msb	most significant bit
PCM	pulse code modulation
PT	packet telemetry
PTDP	packet telemetry data packet
PTFR	packet telemetry data frame
TmNS	Telemetry Network Standard

This page intentionally left blank.

## CHAPTER 7

### Packet Telemetry Downlink

This standard defines the method for transporting variable-length, well-defined data formats in a Chapter 4 pulse code modulation (PCM) stream.

#### 7.1 Packet Telemetry

Packet telemetry (PT) defines the method for asynchronously inserting data from one or more data streams into PCM minor frames. This standard defines various PT packet types that are used to encapsulate well-defined data formats, including:

- Chapter 10 packets;
- Chapter 24 TmNSMessages;
- Ethernet frames.

A PT packet is encapsulated into an integral number of packet telemetry data packets (PTDPs) – nominally each PTDP contains only one PT packet. Different PT packet types may be multiplexed simultaneously into a single, logical stream of PTDPs, which is then segmented into fixed-length packet telemetry data frames (PTFRs) that are inserted into a PCM stream. While a PTFR may be segmented and interspersed with PCM data within a PCM minor frame, each PCM minor frame shall contain only one PTFR. A low-latency PTDP (LLP) mechanism allows for the insertion of one or more PTDPs with low-latency requirements into the transmission of a long PTDP. Specific structure-critical elements are protected using a Golay code; see [Appendix 7-A](#) for details. [Figure 7-1](#) provides an overview of the entire packet telemetry encapsulation process.

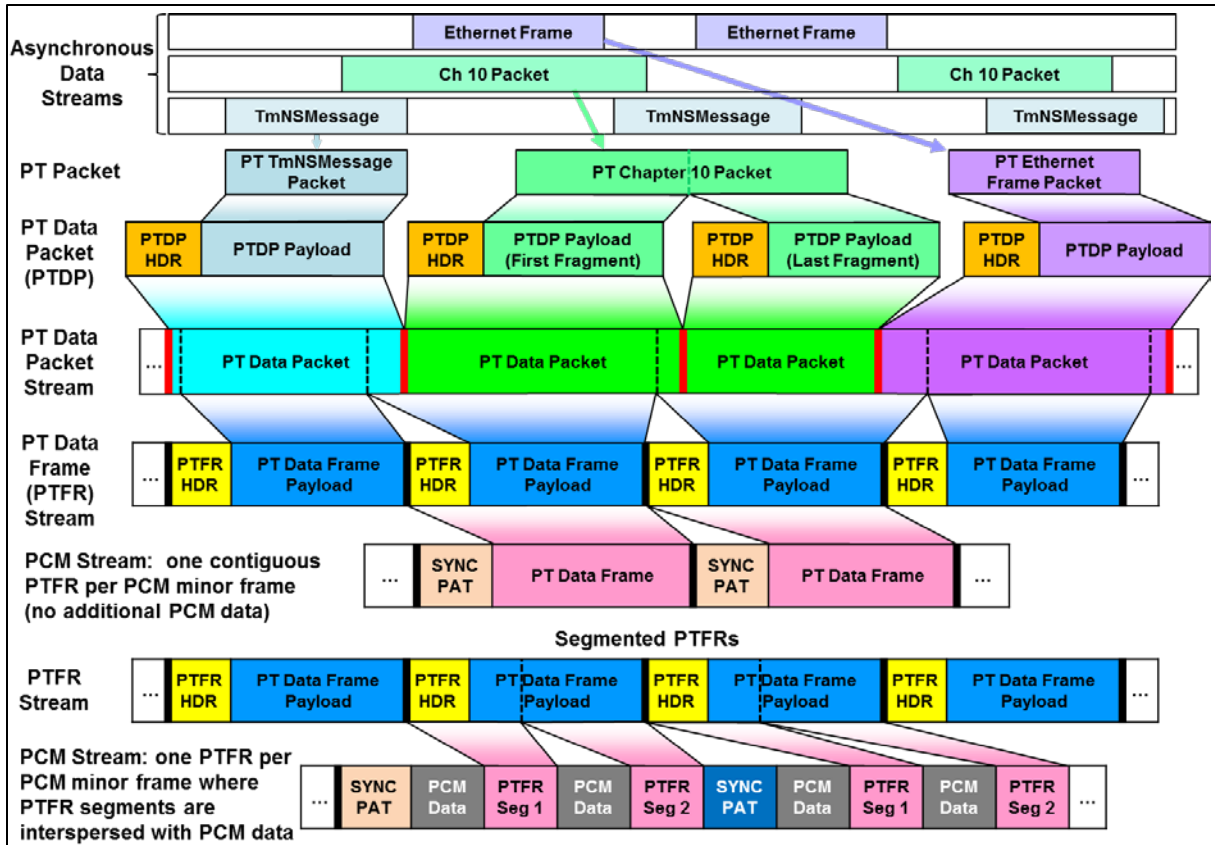





Figure 7-1. Packet Telemetry Overview

**NOTE**  A PT packet may be fragmented into multiple PTDPs where each PTDP's payload contains a fragment of the PT packet. Subsection [7.2.3](#) describes PT packet fragmentation.

A PTFR may be segmented for insertion into a PCM minor frame. A single PCM minor frame may contain multiple PTFR segments. See Section [7.5](#) for more details.

**NOTE**  The IRIG 106-15 restricted a PCM minor frame to a single, complete PTFR. The current standard supports multiple PTFR segments per PCM minor frame.

**NOTE**  [Chapter 9](#) supports defining PTFR segments contained within a PCM minor frame.

## 7.2 Packet Telemetry Data Packet Structure

A PTDP shall include a header and a payload as shown in [Figure 7-2](#).

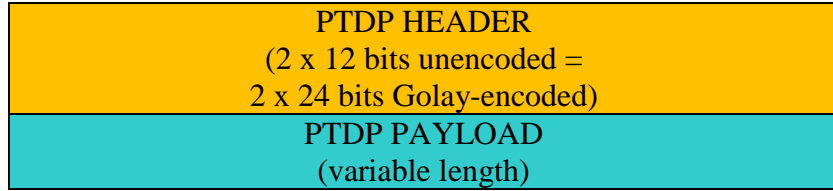


Figure 7-2. PTDP Structure

7.2.1 PTDP Header

The PTDP header shall consist of two 12-bit words and shall be encoded as two 24-bit Golay code words prior to insertion into the PTDP, encoded most significant bit (msb) first and in the word order as shown in [Figure 7-3](#).

	11	10	9	8	7	6	5	4	3	2	1	0
<i>Word 0</i>	Reserved		Content			Fragment		Length (15 – 12)				
<i>Word 1</i>	Length (11 – 0)											

Figure 7-3. PTDP Header Structure (Unencoded)

The PTDP header consists of the following fields.

- Reserved (bits 23 – 22). All bits shall be set to zero (e.g., 2'b00) on transmission; ignored on reception.
- Content (bits 21 – 18). The PT packet type field shall specify the type of PTDP payload (see Subsection [7.2.2](#) for details).
  - 4'b0000: PT Fill Packet
  - 4'b0001: PT Application-Specific Packet
  - 4'b0010: PT Test Counter Packet
  - 4'b0011: PT Chapter 10 Packet
  - 4'b0100: PT Raw Ethernet Media Access Control (MAC) Frame Packet
  - 4'b0101: PT Internet Protocol (IP) Packet
  - 4'b0110: PT Chapter 24 TmNSMessage Packet
  - 4'b0111 –
  - 4'b1111: Reserved
- Fragment (bits 17 – 16). The PT packet fragmentation flags shall identify whether the PTDP payload contains a complete PT packet or a fragment.
  - 2'b00: Complete PT Packet
  - 2'b01: First Fragment of a PT Packet
  - 2'b10: Middle Fragment of a PT Packet
  - 2'b11: Last Fragment of a PT Packet
- Length (bits 15 – 0). The PTDP length field shall provide the length (in bytes) of the PTDP payload (note the PTDP header length is not included in the PTDP length). If

a PT packet exceeds the maximum PTDP length, the PT packet must be fragmented using multiple PTDPs as specified in Subsection [7.2.3](#).

7.2.2 PTDP Payload

The PTDP payload shall contain either a complete PT packet or a PT packet fragment and the Content field identifies the type. The following subsections contain a detailed description for each PT packet type.

7.2.2.1 PT Fill Packet

If no PT packets are available for embedding into a PTDP stream, PT fill packets shall be generated and inserted into the PTDP stream to assure a constant data flow to the PCM stream. Each PT fill packet byte shall be set to 8'b10101010 (0xAA) as shown in [Figure 7-4](#). A PT fill packet size may be an arbitrary integral number of bytes.

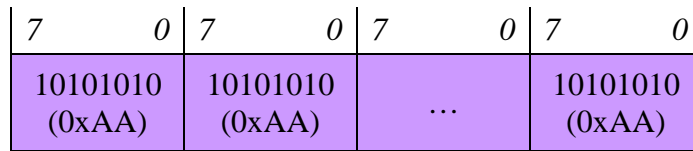


Figure 7-4. PT Fill Packet

7.2.2.2 PT Application-Specific Packet

This standard does not define the format of PT application-specific packets. While PT application-specific packets are allowed, they shall not be used to encapsulate data that conforms to another defined PT packet format (e.g., Chapter 10 packets, Chapter 24 TmNSMessages, Ethernet data, etc.).

7.2.2.3 PT Test Counter Packet

The PT test counter packet is defined as a free-running 12-bit counter. The PT test counter packet shall consist of one 12-bit word and shall be encoded as one 24-bit Golay code word, encoded msb first as shown in [Figure 7-5](#).

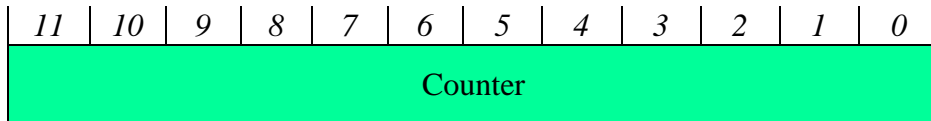



Figure 7-5. PT Test Counter Packet (Unencoded)

 <p><b>NOTE</b></p>	The test counter packet is optional and this standard does not specify the transmission rate.
--	---

7.2.2.4 PT Chapter 10 Packet

The PT Chapter 10 packet structure contains a slightly modified version of a Chapter 10 packet. The primary differences between the original Chapter 10 packet header and the PT Chapter 10 header are:

- The PT Chapter 10 packet does not contain the packet sync pattern field;
- The PT Chapter 10 packet does not contain the packet length field;
- The PT Chapter 10 packet contains a packet trailer bytes field;



- The PT Chapter 10 packet may contain fewer fill bytes than the original Chapter 10 packet header.

Subsection [7.2.2.4.1](#) describes PT Chapter 10 packet composition. Subsection [7.2.2.4.2](#) describes Chapter 10 packet reassembly.

[Figure 7-6](#) shows the Chapter 10 general packet structure and [Figure 7-7](#) shows the PT Chapter 10 packet structure.

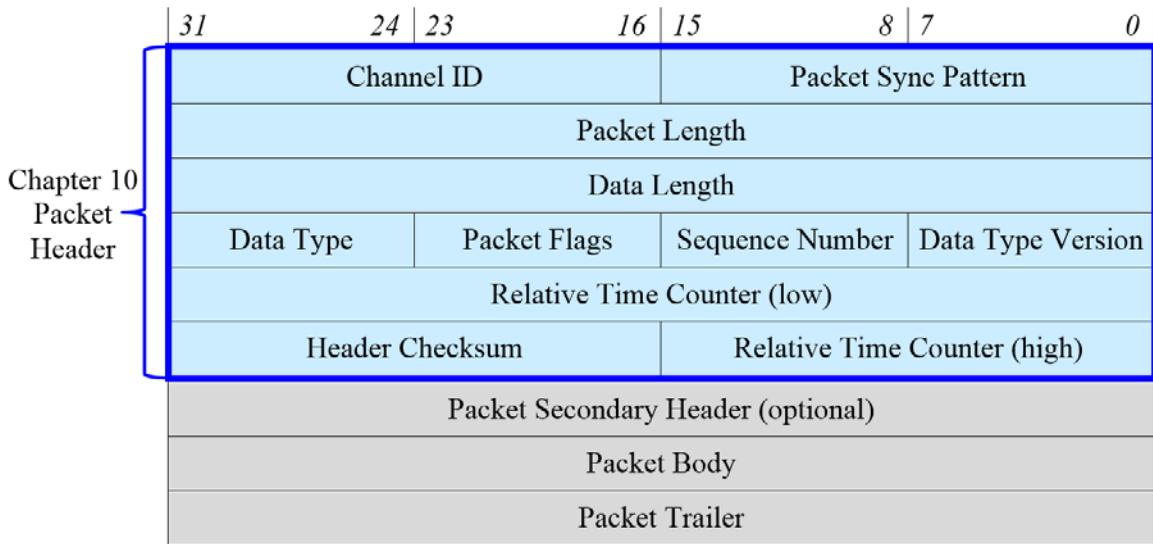


Figure 7-6. Chapter 10 General Packet Structure

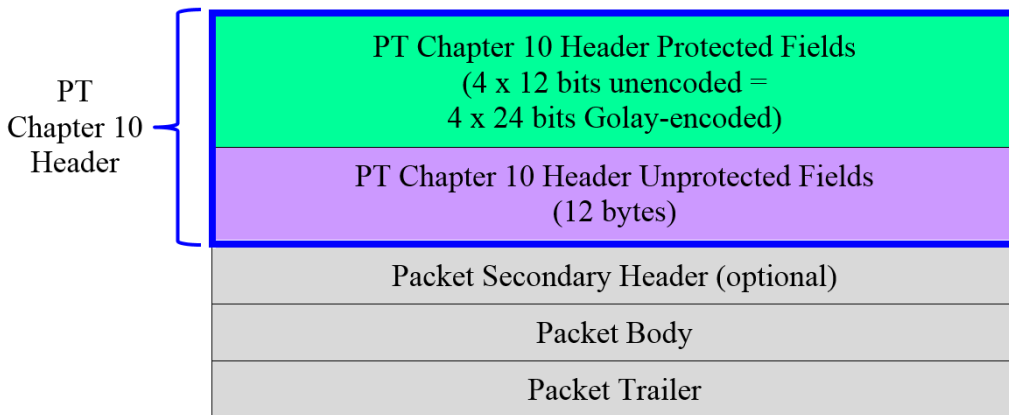


Figure 7-7. PT Chapter 10 Packet Structure

The original Chapter 10 packet header fields are partitioned into two groups for the PT Chapter 10 header:

- Golay-encoded protected fields that protect structure-critical header information;
- Unprotected fields.

The PT Chapter 10 header protected fields shall consist of four 12-bit words and shall be encoded as four 24-bit Golay code words prior to insertion into the PTDP payload, encoded msb first and in the word order indicated in [Figure 7-8](#).

	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	Reserved (0)								Channel ID (15 – 12)			
Word 1	Channel ID (11 – 0)											
Word 2	Packet Trailer Bytes <sup>1</sup> (4 – 0)				Data Length <sup>2</sup> (18 – 12)							
Word 3	Data Length <sup>2</sup> (11 – 0)											
<sup>1</sup> The packet trailer bytes shall contain the sum of the PT Chapter 10 packet’s secondary header length, fill bytes length, and packet checksum length. See Subsection <a href="#">7.2.2.4.1</a> . <sup>2</sup> The Data Length field size limit of 19 bits is sufficient for all Chapter 10 packet sizes, except Computer-Generated Data Packet, Format 1 setup record. See Subsection <a href="#">7.2.2.4.1</a> .												

Figure 7-8. PT Chapter 10 Header Protected Fields (Unencoded)

The PT Chapter 10 header unprotected fields shall consist of 12 bytes as shown in [Figure 7-9](#).

31	24	23	16	15	8	7	0
Data Type		Packet Flags		Sequence Number		Data Type Version	
Relative Time Counter (low)							
Header Checksum				Relative Time Counter (high)			

Figure 7-9. PT Chapter 10 Header Unprotected Fields

**7.2.2.4.1 PT Chapter 10 Packet Composition**


The following steps shall be executed prior to constructing a PT Chapter 10 packet.

1. Truncate the number of packet trailer fill bytes to no more than three bytes. The number of fill bytes contained in a PT Chapter 10 packet shall be restricted to a maximum of three bytes.
2. Update the header packet length. If the packet trailer fill bytes were truncated, the packet length shall be updated accordingly.
3. Calculate a new header checksum. If the packet trailer fill bytes were truncated, the header checksum shall be recalculated using the updated header packet length.
4. Calculate a new data checksum (if the packet trailer contains a data checksum). If the packet trailer fill bytes were truncated and non-zero packet trailer fill bytes were removed, the data checksum shall be recalculated using the truncated packet trailer fill bytes.

Once these steps are executed, the Chapter 10 packet header shall be divided into the protected and unprotected fields as described above. The packet trailer bytes field shall contain

the sum of the PT Chapter 10 packet’s secondary header length, fill bytes length (adjusted to a maximum of three bytes), and data checksum length.

If the size of the Chapter 10 packet exceeds the 19-bit limit, the data length shall be set to modulo 524,288 ( $2^{19}$ ) of the Chapter 10 packet length and multiple Chapter 10 PTDPs shall be generated using the PT packet fragmentation method described in Subsection [7.2.3](#).

 <p><b>NOTE</b></p>	<p>Compared to the original Chapter 10 packet, the resulting PT Chapter 10 packet is either identical, or due to fill byte truncation, shorter than the original Chapter 10 packet. If fill byte truncation occurred, the packet length and packet header checksum will be recalculated and the data checksum may be recalculated; however, the PT Chapter 10 packet’s data content remains unchanged from the original Chapter 10 packet.</p>
--	--

**7.2.2.4.2 Chapter 10 Packet Reassembly**

A Chapter 10 packet may be reassembled once an entire PT Chapter 10 packet has been reassembled from one or more PTDPs – see Subsection [7.2.3](#) for details. The Chapter 10 packet header shall be reassembled after Golay-decoding is performed on the PT Chapter 10 header’s protected fields. The following steps shall be executed.

1. The reassembled Chapter 10 packet sync pattern shall be set to 0xEB25 (16’b1110101100100101) as specified in [Chapter 10](#).
2. The reassembled Chapter 10 packet’s packet length shall be calculated as follows:


$$Packet\ Length = \sum (length\ in\ each\ PTDP\ Fragment\ Header)$$

3. The reassembled Chapter 10 packet’s data length shall be calculated as follows:

$$Data\ Length = calculated\ Packet\ Length - Packet\ Header\ Length\ (24\ bytes) - Packet\ Trailer\ Bytes$$


Perform this comparison to validate the reassembled Chapter 10 packet’s data length:

$$PT\ Chapter\ 10\ Packet's\ Data\ Length = calculated\ Data\ Length\ modulo\ 524,288$$

 <p><b>NOTE</b></p>	<p>The modulo 524,288 (<math>2^{19}</math>) is required to accommodate original Chapter 10 packets that are larger than 524,288 bytes.</p>
--	--

The following steps may be performed to verify the integrity of the reassembled Chapter 10 packet.

1. The packet header checksum should be calculated and compared to the reassembled Chapter 10 packet header checksum.
2. If present, the data checksum in the packet trailer should be calculated and compared to the reassembled Chapter 10 packet data checksum in the packet trailer.

 <p><b>NOTE</b></p>	<p>If fill byte truncation occurred during PT Chapter 10 packet composition, the reassembled Chapter 10 packet length and packet header checksum will differ and the data checksum may differ from the original Chapter 10 packet;</p>
--	--

however, the reassembled Chapter 10 packet’s data content remains unchanged from the original Chapter 10 packet.
--

7.2.2.5 PT Raw Ethernet Media Access Control Frame Packet

The PT raw Ethernet MAC frame packet contains one physical-layer MAC frame, starting with the MAC destination address and ending with the frame check sequence inclusive, as shown in [Figure 7-10](#). The PT raw Ethernet MAC frame packet can contain any kind of message data, IPv4, IPv6, and jumbo messages. No extra protection is applied to the PT raw Ethernet MAC frame packet.

Destination MAC Address 6 bytes	Source MAC Address 6 bytes	LLC (opt) 3 – 9 bytes	Ethertype 2 bytes	Payload 46 – 1500 bytes	Frame Check Sequence 4 bytes
--	-------------------------------------	-----------------------------	----------------------	----------------------------	------------------------------------

Figure 7-10. PT Raw Ethernet MAC Frame Packet

7.2.2.6 PT Internet Protocol Packet

The PT IP packet contains one IPv4 as shown in [Figure 7-11](#) or one IPv6 packet as shown in [Figure 7-12](#). No extra protection is applied to the PT IP packet.

IPv4 Header 20 – 36 bytes	IPv4 Payload 20 – 65,536 bytes
------------------------------	-----------------------------------

Figure 7-11. PT IPv4 Packet

IPv6 Header 40 bytes	IPv6 Payload 40 – 65,536 bytes
-------------------------	-----------------------------------

Figure 7-12. PT IPv6 Packet

7.2.2.7 PT TmNSMessage Packet

The PT TmNSMessage structure contains a slightly modified version of a Chapter 24 TmNSMessage. [Figure 7-13](#) shows the Chapter 24 TmNSMessage structure and [Figure 7-14](#) shows the PT TmNSMessage packet structure.

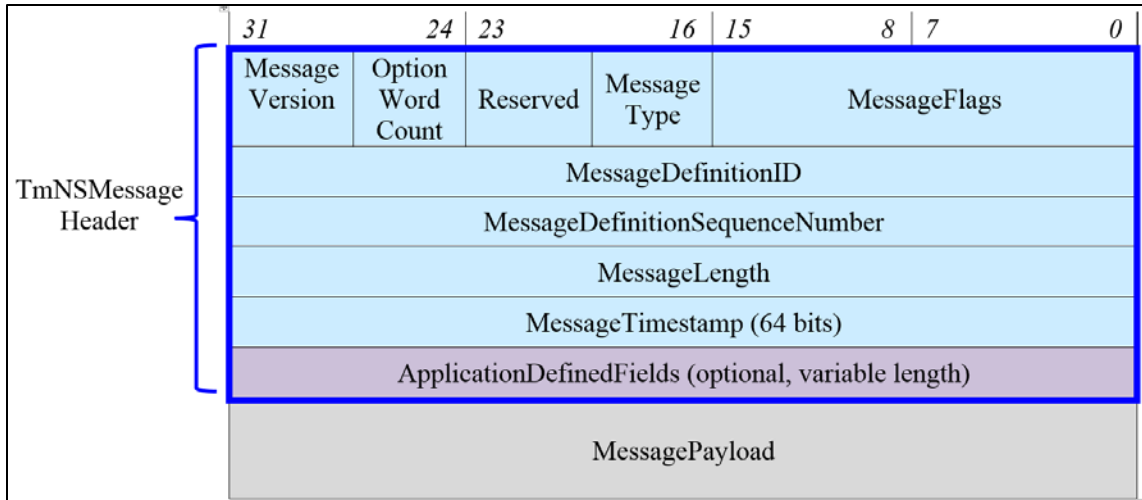


Figure 7-13. Chapter 24 TmNSMessage Structure

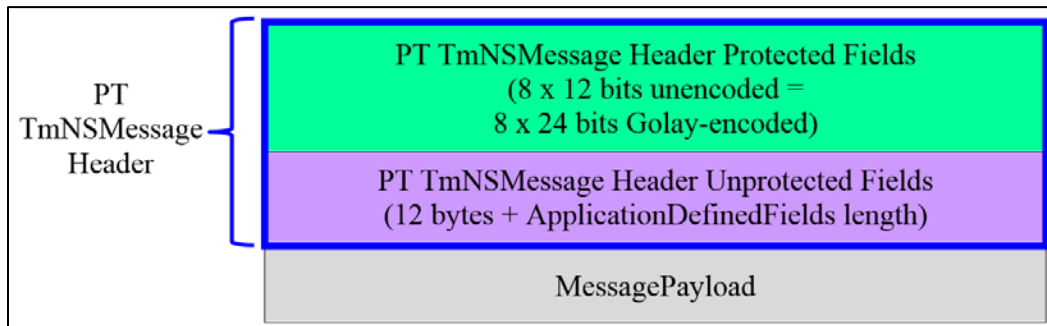


Figure 7-14. PT TmNSMessage Packet Structure

The original Chapter 24 TmNSMessage header fields are partitioned into two groups for the PT TmNSMessage header:

- Golay-encoded protected fields that protect structure-critical header information;
- Unprotected fields (those fields not part of the protected fields).

The PT TmNSMessage header protected fields shall consist of eight 12-bit words and shall be encoded as eight 24-bit Golay code words prior to insertion into the PTDP payload, encoded msb first and in the word order shown in [Figure 7-15](#).

	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	Message Version				OptionWordCount				MessageFlags (3 – 0)			
Word 1	MessageFlags (15 – 4)											
Word 2	Reserved				MessageDefinitionID (31 – 24)							
Word 3	MessageDefinitionID (23 – 12)											
Word 4	MessageDefinitionID (11 – 0)											
Word 5	MessageType				MessageLength (31 – 24)							
Word 6	MessageLength (23 – 12)											
Word 7	MessageLength (11 – 0)											

Figure 7-15. PT TmNSMessage Header Protected Fields (Unencoded)

The PT TmNSMessage header unprotected fields shall consist of 12 bytes plus the ApplicationDefinedFields (if present) as shown in [Figure 7-16](#).

31	24	23	16	15	8	7	0
MessageDefinitionSequenceNumber							
MessageTimestamp (64 bits)							
ApplicationDefinedFields (optional, variable length)							

Figure 7-16. PT TmNSMessage Header Unprotected Fields

### 7.2.3 PT Package Fragmentation

If a PT packet requires fragmentation, each PTDP containing a PT packet fragment shall be inserted sequentially into the PTFR stream. Only LLPs can be inserted in between a PT packet’s sequence of fragments by using the LLP encapsulation mechanism as described in Subsection [7.4.2](#). While fragmentation is necessary if a PT packet’s size is greater than or equal to 64 kilobytes, any PT packet may be fragmented. [Figure 7-17](#) shows PT packet fragmentation and reassembly.

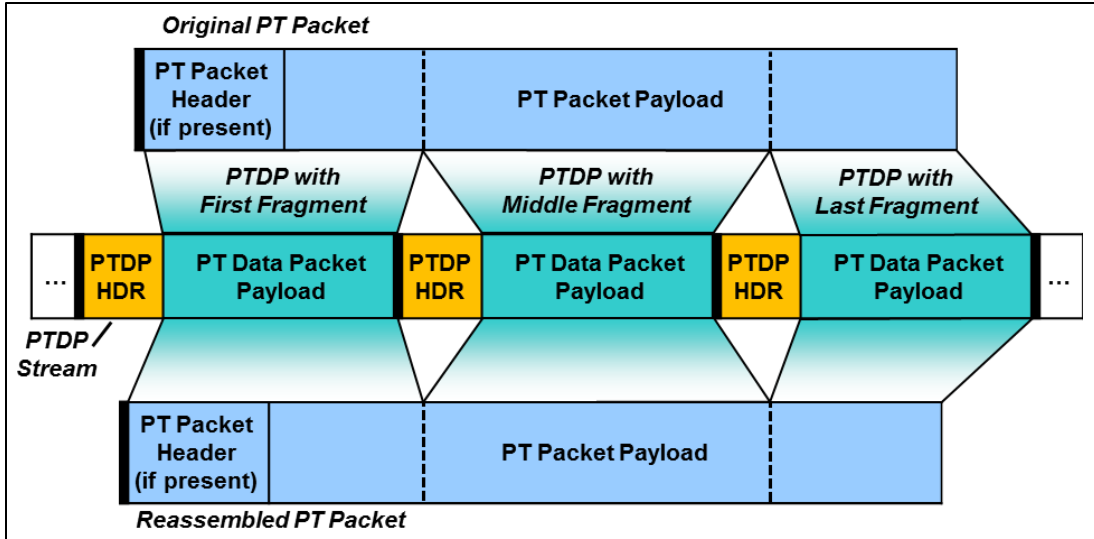


Figure 7-17. PT Packet Fragmentation and Reassembly

### 7.3 Packet Telemetry Data Frame Structure

The PTFR is a fixed-length frame of data that contains a portion of a continuous PTDP stream as represented in [Figure 7-18](#).

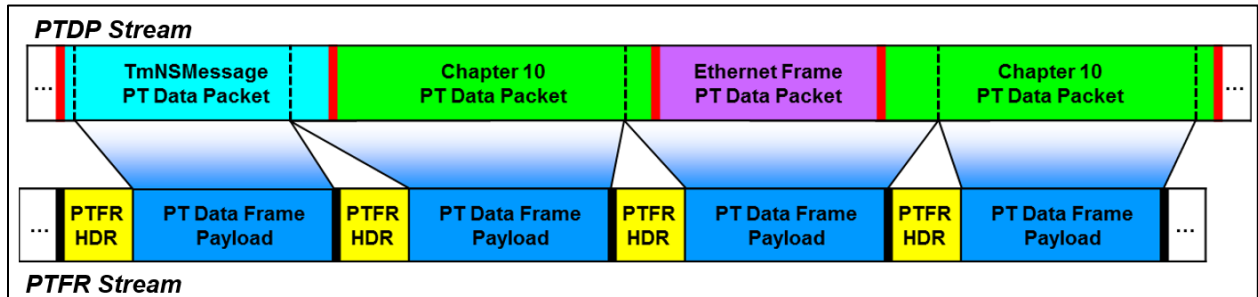


Figure 7-18. Packet Telemetry Data Frame (PTFR) Overview

A PTFR consists of a header and a payload as shown in [Figure 7-19](#).

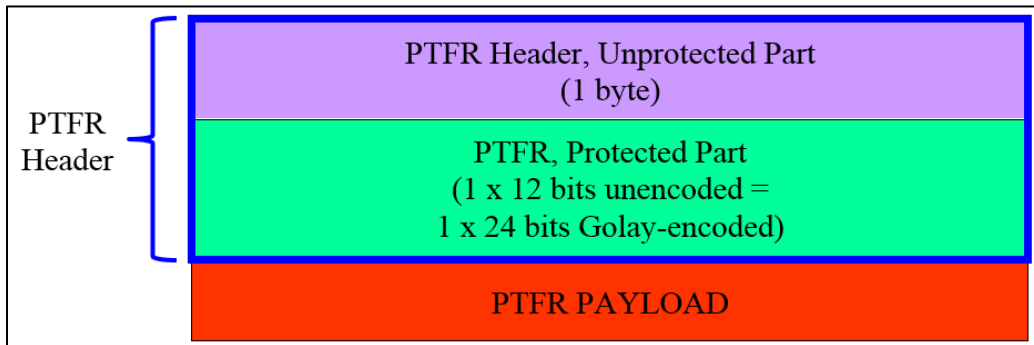


Figure 7-19. PT Data Frame Structure

#### 7.3.1 PTFR Header

The PTFR header fields are partitioned into two groups:

- Unprotected fields that contains stream and version information;
- Golay-encoded protected fields that protect structure-critical header information.

### 7.3.1.1 PTFR Header Unprotected Fields

The PTFR header unprotected fields shall consist of one byte as shown in [Figure 7-20](#).

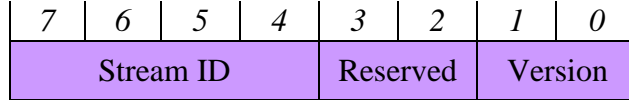


Figure 7-20. PTFR Header Unprotected Fields

The PTFR header unprotected fields are defined below.

- Stream ID (bits 7 – 4). The stream ID identifies an application-specific stream.
- Reserved (bits 3 – 2). All bits shall be set to zero (e.g., 2'b00) on transmission; ignored on reception.
- Version (bits 1 – 0). The PTFR version:
  - 2'b00: Version 1
  - 2'b01: Reserved
  - 2'b10: Reserved
  - 2'b11: Reserved

### 7.3.1.2 PTFR Header Protected Fields

The PTFR header protected fields shall consist of one 12-bit word and shall be encoded as one 24-bit Golay code word prior to insertion into the PTDP payload, encoded msb first and in the word order indicated in [Figure 7-21](#).

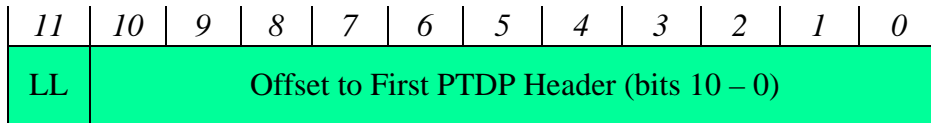


Figure 7-21. PTFR Header Protected Fields (Unencoded)

The PTFR header protected fields are defined below.

- LL: LLP Exists (bit 11) – see Subsection [7.4.2](#) for LLP details
  - 1'b1: indicates the PTFR payload contains one or more optional LLPs and the closing LLP end byte.
  - 1'b0: indicates no LLP and no LLP end byte exist in the PTFR payload.
- Offset to First PTDP Header (bits 10 – 0). If a PTFR contains at least one PTDP header, this field specifies a byte offset to the first byte of that first PTDP. Otherwise, all bits shall be set to one (11'b1111111111). The value is relative to the first data byte following the PTFR header (e.g., the value of zero represents the first byte following the PTFR header). See Subsection [7.4.1](#) for additional information.



### 7.3.2 PTFR Payload

The PTFR payload contains zero or more partial or complete PTDPs as illustrated in [Figure 7-22](#). Optional LLPs may be inserted in the PTFR payload after the PTFR header (see Subsection [7.4.2](#) for LLP details).

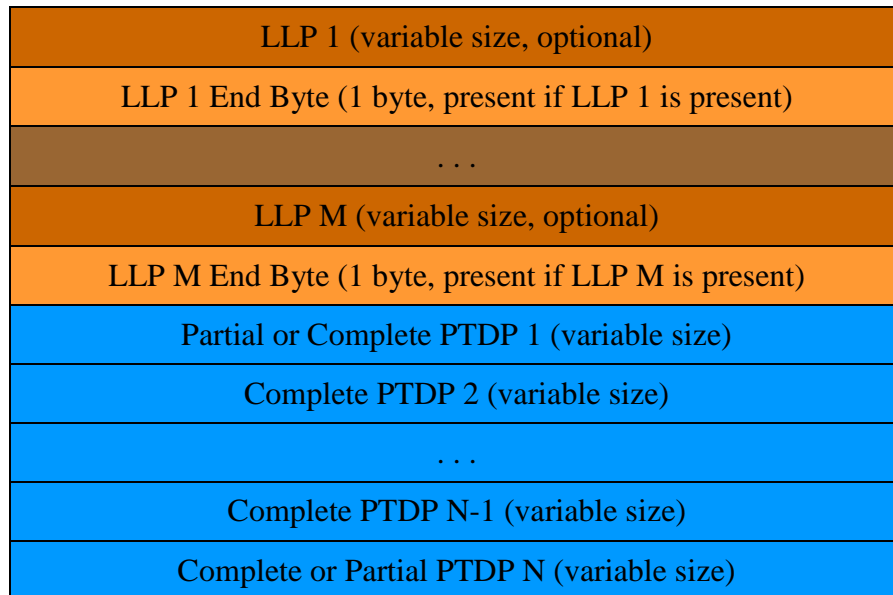


Figure 7-22. PTFR Payload Structure

#### 7.3.2.1 Low-Latency PTDP

An LLP is a PTDP having low-latency transmission requirements as described in Subsection [7.4.2](#). If one or more LLPs exist in a PTFR, the first LLP is placed immediately after the PTFR header.

#### 7.3.2.2 Low-Latency PTDP End Byte

For each LLP contained within a PTFR, an LLP end byte shall immediately follow the LLP. The LLP end byte identifies if another LLP follows or if this is the last LLP in the PTFR. The LLP end byte encoding is as follows.

- 8b'11111111 (0xFF) indicates that another LLP immediately follows this byte.
- 8b'00000000 (0x00) indicates there are no more LLPs in this PTFR. The byte following this end byte is the first byte of the first PTDP, unless the LLP end byte is the PTFR's last byte (i.e., there are no PTDPs in the PTFR's payload).

#### 7.3.2.3 PTDPs in PTFR Payload

The PTFR payload contains zero or more partial or complete PTDPs as illustrated in [Figure 7-22](#). The initial and last PTDPs may be either partial or complete PTDPs. See Subsection [7.4.1](#) for details.

## 7.4 Asynchronous PTDP Multiplexing

A PTFR contains asynchronously inserted PTDPs; one PTDP may span multiple PTFRs. The PTDP stream contains a continuous series of PTDPs; the start of a PTDP must immediately follow the last byte of a previous PTDP as illustrated in [Figure 7-23](#).

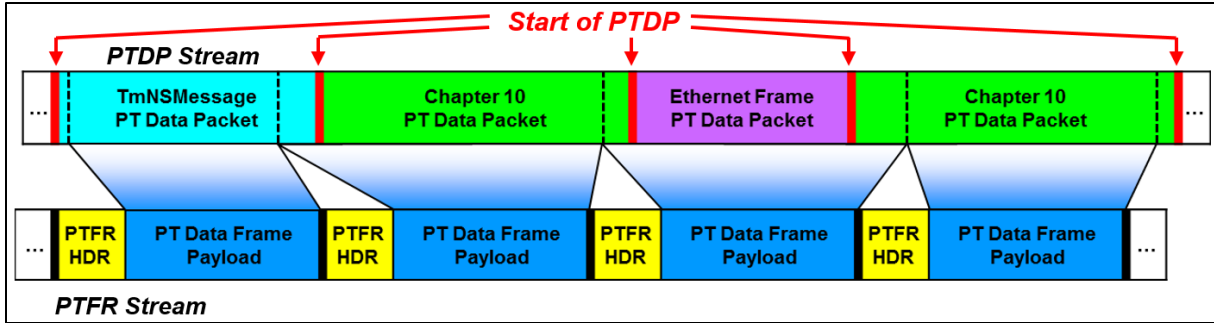


Figure 7-23. Start of PTDPs Overview

#### 7.4.1 Standard PTDP Encapsulation

If the start of a PTDP is contained within a PTFR, the PTFR header shall contain the offset to the PTDP’s first byte. If there are multiple PTDPs in the PTFR, the PTFR header shall contain the offset to the start of the first PTDP. Since one PTDP may span multiple PTFRs, the PTFR header may not contain an offset to a PTDP. See Subsection 7.3.1.2 for details. An overview of the standard PTDP encapsulation mechanism is shown in Figure 7-24.

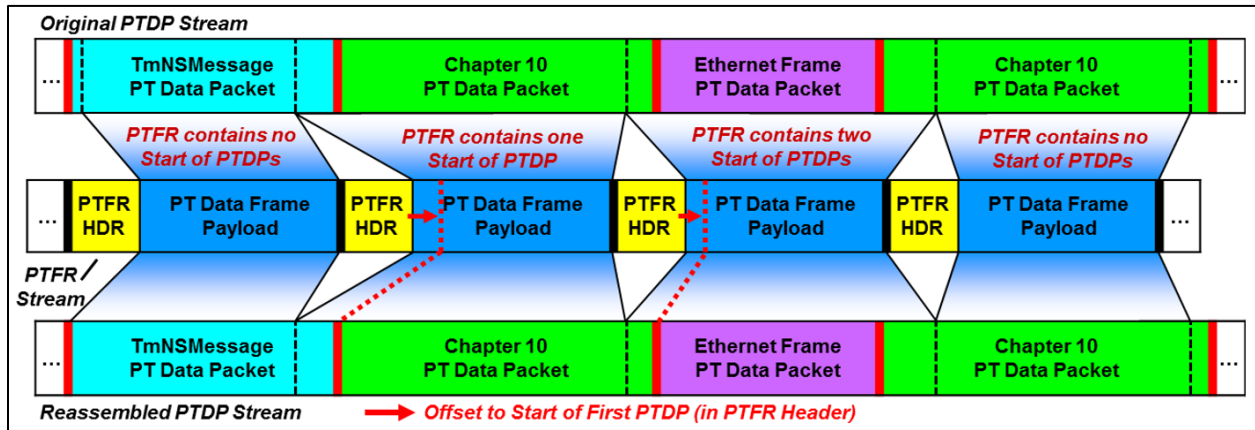


Figure 7-24. Standard PTDP Encapsulation Overview

#### 7.4.2 Low-Latency PTDP Encapsulation

The transmission of a long PTDP may cause too long of latency for some critical data. Therefore, an LLP mechanism is provided, allowing the insertion of one or more PTDPs with low-latency requirements within the transmission of a long PTDP. The interrupted long PTDP is resumed immediately after the LLP part of the PTFR.

An LLP shall not span multiple PTFRs. When constructing a PTFR, an entire LLP and associated end byte shall fit into the remaining space in the PTFR. Figure 7-25 shows an overview of PTDP encapsulation with LLPs.

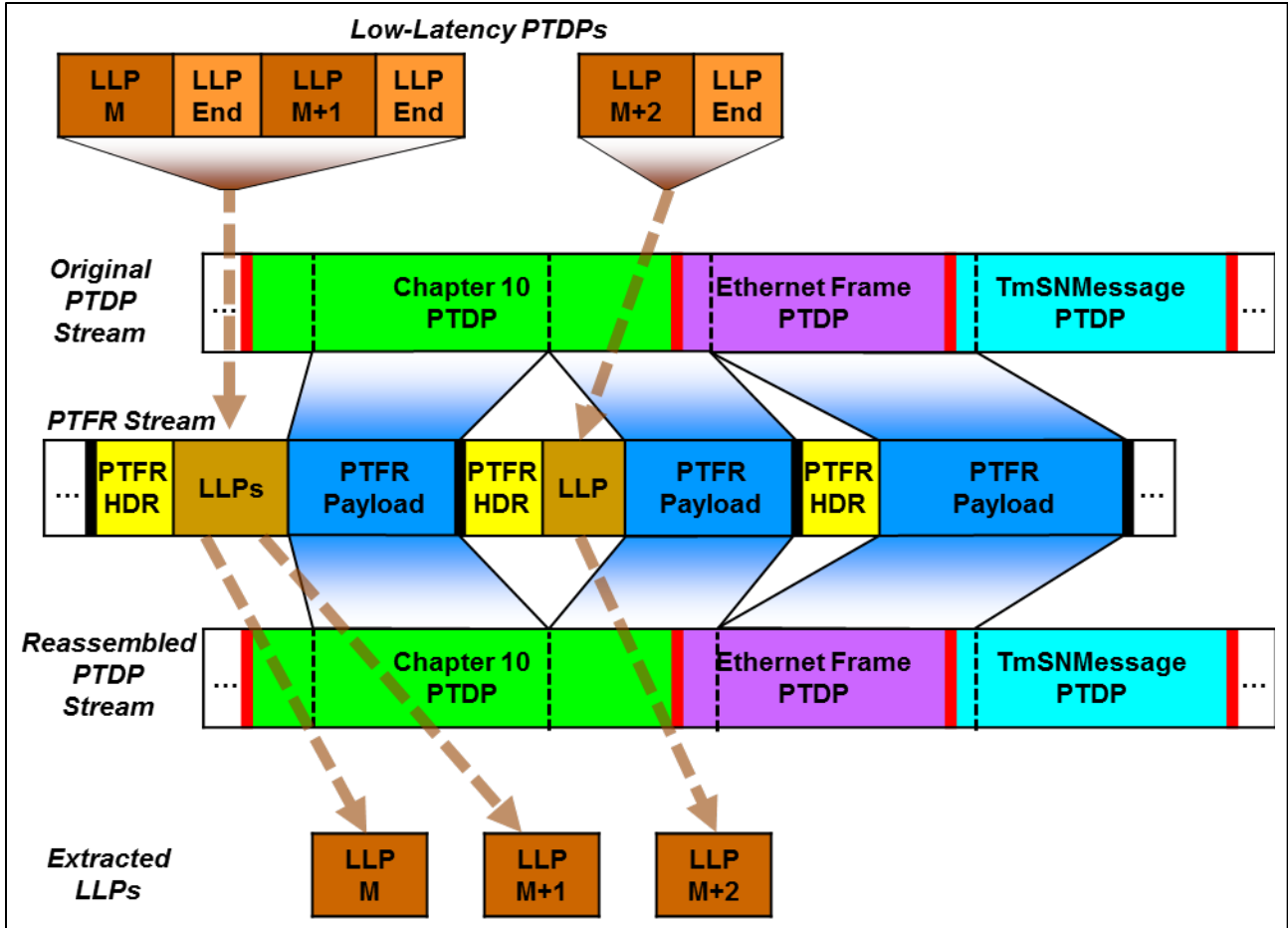



Figure 7-25. PTDP Encapsulation with LLPs Overview

**NOTE**  The offset to the first PTDP in the PTFR header is not necessarily pointing immediately after the LLP.

## 7.5 PT Data Frame Transport

To insert a PTFR into a Chapter 4 PCM minor frame, each PTFR segment is byte-aligned and inserted into the PCM minor frame as a byte stream with the msb first (as shown in [Figure 7-26](#)). If a PTFR segment's size is not an integral number of bytes, the remaining bits at the end of the PTFR segment are considered fill bits and should be ignored. Each PCM minor frame shall contain exactly one complete PTFR structure; [Figure 7-27](#) shows a PCM minor frame with two PTFR segments.

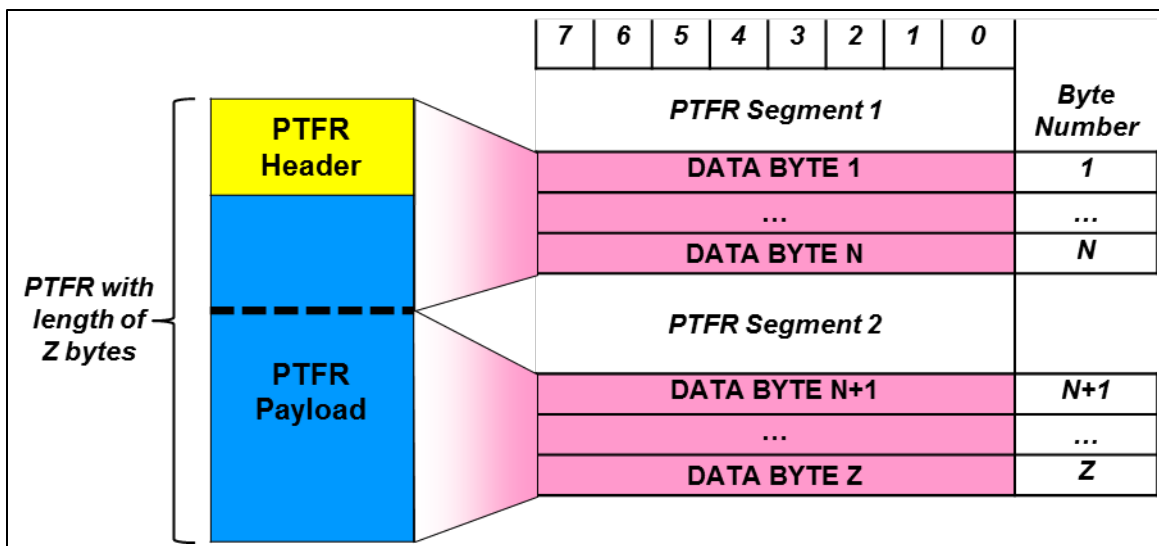


Figure 7-26. Splitting a PTFR into Two Segments

**NOTE** Any PTFR segmentation does not affect how the PTFR header's offset to first PTDP header is used to index into the PTFR byte array.

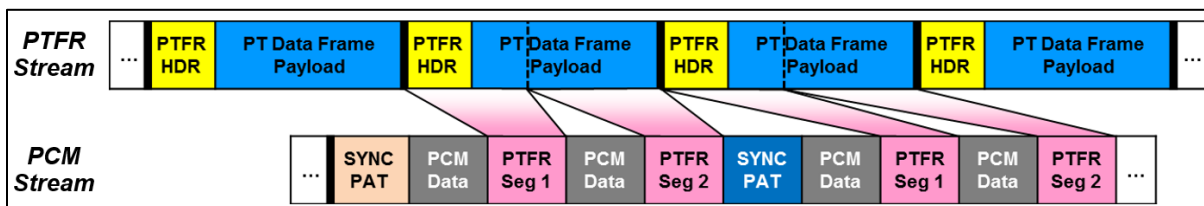


Figure 7-27. PCM Minor Frame With Two PTFR Segments

If no PTDPs are available for transmission, PTDPs with PT fill packets shall be inserted into the PTFR for subsequent insertion into the PCM minor frame.

**NOTE** All PCM minor frame overhead words such as the Chapter 4 sync pattern or subframe counters are not considered part of a PTFR.

## APPENDIX 7-A

## Extended Binary Golay Code

## A.1. Introduction

A single-bit transmission error may cause excessive data loss in packet telemetry. If the error occurs in identification or structure length fields, the error can lead to misinterpretation of a packet or a loss of a series of packets.

To help mitigate potential errors, a self-correcting coding called extended binary Golay code is applied to structure-critical elements in a packet telemetry stream. This additional coding provides protection of the packet identification and length information and supports correction of up to 3-bit transmission errors in a 24-bit sequence. This is accomplished by encoding 12-bit words into 24-bit words.

This extended binary Golay code,  $G_{24}$  (sometimes just called the “Golay code” in finite group theory) encodes 12 bits of data in a 24-bit word in such a way that any 3-bit errors can be corrected or any 7-bit errors can be detected. In standard code notation the codes have parameters (24, 12, 8) corresponding to the length of the code words, the dimension of the code, and the minimum Hamming distance between two code words, respectively.<sup>1</sup> The coding and decoding of the Golay code is illustrated in [Figure A-1](#).

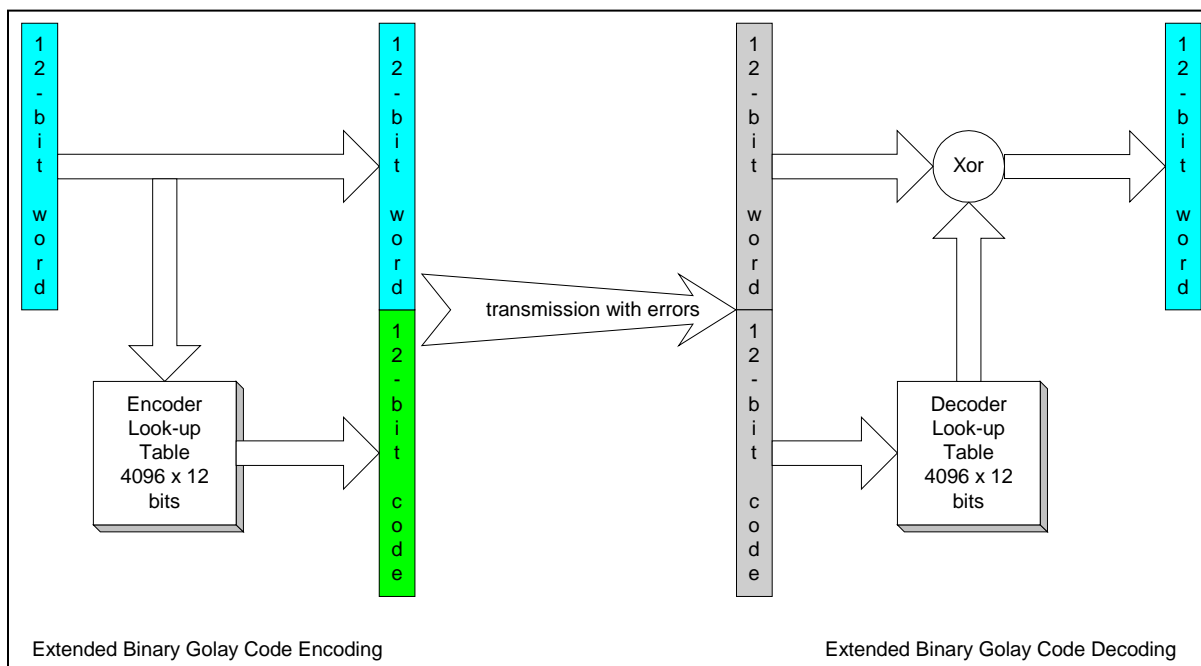


Figure A-1. Golay Code Encoding and Decoding

The following sections are C code reference implementation and define the required behavior of encoding and decoding the extended binary Golay code.

<sup>1</sup> Golay, Marcel J. E. *Notes on Digital Coding* in “Proceedings of the IRE,” 1949, v.37, p. 657.

## A.2. Encoding Golay Code

The extended binary Golay code encoding lookup table can be initialized by the `InitGolayEncode()` function, and the encoding can be done by the `Encode(v)` macro of the following C code.

```
#define GOLAY_SIZE      0x1000

// Generator matrix : parity sub-generator matrix P :

uint16_t G_P[12] = {
    0xc75, 0x63b, 0xf68, 0x7b4,
    0x3da, 0xd99, 0x6cd, 0x367,
    0xdc6, 0xa97, 0x93e, 0x8eb
};

/* Binary representation
 1 1 0 0   0 1 1 1   0 1 0 1
 0 1 1 0   0 0 1 1   1 0 1 1
 1 1 1 1   0 1 1 0   1 0 0 0
 0 1 1 1   1 0 1 1   0 1 0 0
 0 0 1 1   1 1 0 1   1 0 1 0
 1 1 0 1   1 0 0 1   1 0 0 1
 0 1 1 0   1 1 0 0   1 1 0 1
 0 0 1 1   0 1 1 0   0 1 1 1
 1 1 0 1   1 1 0 0   0 1 1 0
 1 0 1 0   1 0 0 1   0 1 1 1
 1 0 0 1   0 0 1 1   1 1 1 0
 1 0 0 0   1 1 1 0   1 0 1 1
*/
uint32_t      EncodeTable[ GOLAY_SIZE ];      // Golay encoding table

// encode a 12-bit word to a 24-bit Golay code word
#define Encode(v) EncodeTable[v&0xffff]

// initialize the Golay encoding lookup table
void InitGolayEncode( void )
{
    for( uint32_t x=0; x < GOLAY_SIZE; x++ ) {
        // generate encode LUT
        EncodeTable[x]=(x<<12);
        for( uint32_t i=0; i<12; i++ ) {
            if( (x>>(11-i)) & 1 )
                EncodeTable[x] ^= G_P[i];
        }
    }
}
```

## A.3. Decoding Golay Code

The extended binary Golay code decoding lookup tables can be initialized by the `InitGolayDecode()` function of the following C code. The 12-bit decoded and corrected word can be calculated by the `Decode(v)` macro from a 24-bit code word. The number of error bits in a 24-bit code word can be gotten by the `Error(v)` macro from a 24-bit code word.

```

#define      GOLAY_SIZE          0x1000

uint16_t SyndromeTable[ GOLAY_SIZE ];    // Syndrome table
uint16_t CorrectTable[ GOLAY_SIZE ];     // correction table
uint8_t  ErrorTable[ GOLAY_SIZE ];       // number of error bits table

#define Syndrome2(v1,v2)      (SyndromeTable[v2]^(v1))
#define Syndrome(v)          Syndrome2(((v)>>12)&0xfff,(v)&0xfff)
#define Errors2(v1,v2)       ErrorTable[Syndrome2(v1,v2)]
#define Decode2(v1,v2)       ((v1)^CorrectTable[Syndrome2(v1,v2)])

// get the number of error bits in this 24-bit code word
#define Errors(v)             Errors2(((v)>>12)&0xfff,(v)&0xfff)

// get the 12-bit corrected code from a 24-bit code word
#define Decode(v)             Decode2(((v)>>12)&0xfff,(v)&0xfff)

// Parity Check matrix
uint16_t H_P[12] = {
    0xa4f, 0xf68, 0x7b4, 0x3da,
    0x1ed, 0xab9, 0xf13, 0xdc6,
    0x6e3, 0x93e, 0x49f, 0xc75
};

/* Binary representation
  1 0 1 0   0 1 0 0   1 1 1 1
  1 1 1 1   0 1 1 0   1 0 0 0
  0 1 1 1   1 0 1 1   0 1 0 0
  0 0 1 1   1 1 0 1   1 0 1 0

  0 0 0 1   1 1 1 0   1 1 0 1
  1 0 1 0   1 0 1 1   1 0 0 1
  1 1 1 1   0 0 0 1   0 0 1 1
  1 1 0 1   1 1 0 0   0 1 1 0

  0 1 1 0   1 1 1 0   0 0 1 1
  1 0 0 1   0 0 1 1   1 1 1 0
  0 1 0 0   1 0 0 1   1 1 1 1
  1 1 0 0   0 1 1 1   0 1 0 1
*/

// calculate the number of 1s in a 24-bit word
uint8_t OnesInCode( uint32_t code, uint32_t size )
{
    uint8_t ret = 0;
    for( uint32_t i=0; i<size; i++ ) {
        if( (code>>i) & 1 )
            ret++;
    }
    return ret;
}

void InitGolayDecode( void )
{
    for( uint32_t x=0; x < GOLAY_SIZE; x++ ) {
        // generate syndrome LUT

```

```

SyndromeTable[x]=0;          // Default value of the Syndrome LUT
for( uint32_t i=0; i<12; i++ ) {
    if( (x>>(11-i)) & 1 ) SyndromeTable[x] ^= H_P[i];
    ErrorTable[x] = 4;
    CorrectTable[x]=0xffff;
}
}

// no error case
ErrorTable[0] = 0;
CorrectTable[0]= 0;
// generate all error codes up to 3 ones
for( int i=0; i<24; i++ ) {
    for( int j=0; j<24; j++ ) {
        for( int k=0; k<24; k++ ) {
            uint32_t error = (1<<i) | (1<<j) | (1<<k);
            uint32_t syndrome = Syndrome(error);
            CorrectTable[syndrome] = (error>>12) & 0xffff;
            ErrorTable[syndrome] = OnesInCode(error,24);
        }
    }
}
}
}

```

#### A.4. Decoding the Golay Code (8,1,3)

The one-byte 0x00 or 0xff can also be considered as a binary Golay code (8,1,3). It allows correcting the 0x00 or 0xff transmission of up to 3-bit errors, and detecting 4-bit errors. The (8,1,3) code decoding lookup tables shall be initialized by the InitGolay00FFDecode() function, and the decoding can be done by the Decode00FF(v) macro of the following C code.

```

#define      BYTE_LUT_SIZE          0x100

uint8_t Decode00FFTable[ BYTE_LUT_SIZE ]; // decode 0x00 or 0xff (8,1,3)
uint8_t Error00FFTable[ BYTE_LUT_SIZE ]; // number of error bits (8,1,3)

#define Decode00FF(v)      Decode00FFTable[v]
#define Error00FF(v)      Error00FFTable[v]

void InitGolay00FFDecode ( void )
{
    // generate (8,1,3) tables
    for( uint32_t i=0; i<BYTE_LUT_SIZE; i++ ) {
        uint32_t j = OnesInCode(i,8);
        Decode00FFTable[i] = j <= 4 ? 0 : 0xff;
        Error00FFTable[i] = j <= 4 ? j : 8-j;
    }
}

```



## APPENDIX 7-B

### **Citations**

Golay, Marcel J. E. “Notes on Digital Coding” in *Proceedings of the IRE*, 1949, v.37, p. 657.

**\*\*\* END OF CHAPTER 7 \*\*\***