

**EDITOR'S NOTE: The following Chapter 10 and section 18 of Chapter 6 of the IRIG Standard 106-03, *Telemetry Standard*, have been extracted for separate publication at the request of the RCC Telemetry Group pending the full update and re-publication of the entire standard. This document will be withdrawn upon publication of the entire 106-03 revision.**

## **CHAPTER 10**

### **SOLID STATE ON-BOARD RECORDER STANDARD**

#### **10.1 General**

A large number of unique and proprietary data structures have been developed lately for specific data recording applications that require unique decoding software programs. Writing unique decoding software, checking the software for accuracy, and decoding the data tapes is extremely time consuming and costly. In addition, the test ranges have seen the implementation of non-tape-based, high data-rate recorders in the late 1990s — the most predominate being solid-state memory devices. As high-rate digital recorders were fielded, and with solid state on the horizon, the Telemetry Group (TG) formed an ad-hoc committee to research and write a computer compatible, digital data acquisition standard.

It was determined that a solid-state digital data acquisition and on-board recorder standard (see Figure 10-1) would support a broad range of requirements, including the following:

- a. Data download and interface
- b. One or more multiplexed data streams
- c. One or more single data streams
- d. Read-after-write and read-while-write options
- e. Data format definitions
- f. Recorder control
- g. Solid-state media declassification

Specifically, this digital data acquisition standard shall be compatible with the multiplexing of both synchronous and asynchronous digital inputs such as pulse code modulation (PCM) and MIL-STD-1553 data bus, time, analog, video, ARINC 429, discrete, and RS-232/422 communication data. This solid-state recorder standard will allow the use of a common set of playback/data reduction software to take advantage of emerging random access recording media.

The purpose of this Chapter is to establish a common interface standard for the implementation of solid-state digital data acquisition and on-board recording systems by the

organizations participating in the Range Commanders Council (RCC). This standard does not specify hardware architecture, e.g., the coupling of data acquisition, multiplexing, and media storage.

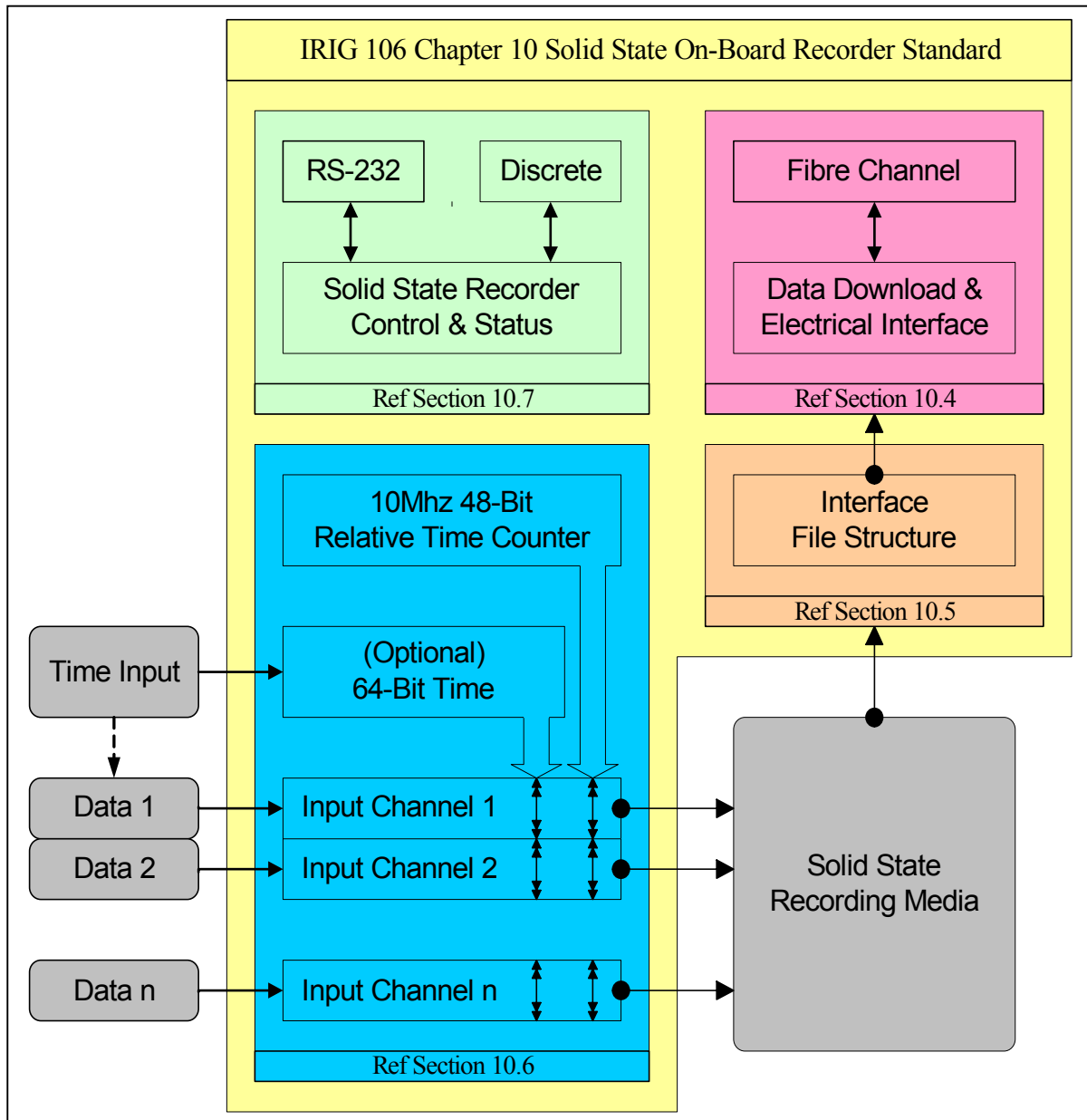


Figure 10-1. Functional layout of standard.

Four interface levels are provided in this standard:

1. Data download and electrical interface, which is the physical interface for data access (defined in section [10.4](#)).
2. Interface file structure, which defines the data access structure (defined in section [10.5](#)).
3. Data format definition, which defines data types and packetization requirements (defined in section [10.6](#)).
4. Solid-state recorder control and status, which defines command and control mnemonics, status, and their interfaces (defined in section [10.7](#)).

## 10.2 **Definitions**

**Bad Block**: Block that has been determined to be unreliable for storing user data.

**Bad Block Table**: Table of bad block entries for a memory board. The data stored in the entry identifies the chip and block number of the bad block. The table entry also contains a flag field. The flag field is used to determine the circumstance in which the bad block was detected. It also provides a flag indicating whether the corresponding bad block has previously been “secure erased.”

**Block**: Storage unit within the flash device. A block is the smallest unit of memory that can be erased.

**Byte**: A contiguous set of 8 bits that are acted on as a unit.

**Channel ID**: All channels in a system must have a unique value (data channels and playback channels).

**Channel Specific Data Words**: A set of required words for a data type channel that has data specific information.

**Checksum**: Arithmetic sum of data bytes or words.

**Erasing Flash**: Performing an erase function on a flash device. Erasing a flash device sets all bits to a known logic state.

**EVPD**: Enable vital product data.

**Intra-Packet Data Header**: Contains time and status information for the tagging of data inside a packet.

**Long Word**: A contiguous set of 32 bits that are acted on as a unit.

**lsb**: The least significant bit of a series of bits.

**LSB**: The least significant byte of a series of bytes.

**LSW**: The least significant word of a series of words.

**LSLW**: The least significant long word of a series of long words.

**Magic Number**: An identifier for the directory block. This is a value chosen to support discovery of lost directory entries and directory reconstruction after a fault.

**Memory Board**: Printed circuit board containing flash memory devices used to store user data.

**msb**: The most significant bit of a series of bits.

**MSB**: The most significant byte of a series of bytes.

**MSW**: The most significant word of a series of words.

**MSLW**: The most significant long word of a series of long words.

**Non-volatile**: Memory media that retains data when power is removed.

**Recorder**: The entity that includes the input and control interfaces, RMM, and functionality required to properly record data.

**Recording Session**: Time interval from first data packet generated to end of the recording.

**Relative Time Counter**: A free-running 10 MHz binary counter represented by 48-bits common to all data channels. The counter shall be derived from an internal crystal oscillator and shall remain free running during each recording session. The applicable data bit to which the 48-bit value applies will be defined in each data type section.

**Removable Memory Module (RMM)**: That element of the data recorder that contains the stored data.

**Packet**: Encapsulates a block of observational and ancillary application data that is to be recorded.

**Packet Header**: Identifies the source and characteristics of the data packet and encapsulation environment.

**Packet Secondary Header**: Contains packet header time.

**Page**: Storage unit within the flash device. A page is the smallest storage unit that can be written.

**Quad Word**: A contiguous set of 64 bits that are acted on as a unit.

**Word**: A contiguous set of 16 bits that are acted on as a unit.

### 10.3 Operational Requirements

This section of the standard specifies the basic operation and required interfaces for the Solid-State Data Storage and Download.

10.3.1 Required Configuration. Every recorder, as a minimum, shall provide the following functionality:

- a. Download port
- b. Control port
- c. External power port

The required download port interface shall be Fibre Channel. This combination will allow data extraction and transfer from any solid-state recorder to any RCC compliant, intermediate storage unit.

10.3.2 Exclusions to this Standard. The physical size, configuration, and form factor for the recorder and/or the RMM are not controlled by this standard. Due to the variation in capacity/rate/cost requirements of the users, this standard does not specify the technology to be used in the RMM or the recorder.

10.3.3 Internal System Management. Any processing performed on the stored data by the recorder (e.g. for the purposes of internal system management, error detection and correction (EDAC), physical frame formatting, etc.) shall be removed from the stored data when the stored data is downloaded or transferred from storage media.

10.3.4 Data Download. The data acquisition recorder may have a removable memory capability or the whole recorder can be removed from the acquisition platform and taken to a ground station for data download. Reference paragraph [10.4.1](#) for electrical interface requirement.

10.3.5 Data Download File Extension. Upon data download to a host computing platform, all IRIG 106 Chapter 10-compliant recordings shall use the file extension, **\*.ch10**. The use of this standard extension will indicate that any file on a ground computing or storage platform is in compliance with section 10.6 of this standard.



**IMPORTANT:** Upon data download to a host computing platform, all IRIG 106 Chapter 10-compliant recordings shall use the file extension, **\*.ch10**.

## 10.4 Data Download and Electrical Interface

In accordance with (IAW) paragraph 10.3.1, the required download port interface shall be Fibre Channel (FC). Physical, signaling, and command protocols contained in paragraphs 10.4.1 and 10.4.2 were adapted from the North Atlantic Treaty Organization (NATO) Military Agency for Standardization (MAS) Standardization Agreement (STANAG) NATO Advanced Data Storage Interface (NADSI) number 4575 hereinafter referred to in this document as STANAG 4575.

10.4.1 Physical and Signaling Protocols. The interface shall comply with FC-PI (physical interfaces) and FC-FS (framing and signaling) with configuration options as specified below.

- a. Physical media. Either Fibre Channel copper or optical fiber interface can be utilized.
- b. Signaling rate. The transmission signaling rate shall be 1.0625 giga-baud.

10.4.2 Command Protocol. The interface shall conform to the requirements of the Fibre Channel - Private Loop SCSI Direct Attach (FC-PLDA)<sup>1</sup> interoperability. Table 17 of FC-PLDA specifies a control protocol using a subset of commands, features, and parameters defined for the Small Computer System Interface (SCSI-3). Table 17 also defines the command feature and parameter usage categories of “Required,” “Allowed,” “Invokable,” and “Prohibited” between the SCSI Initiator and Target. These definitions assume that the Target is a magnetic disk drive or equivalent device.

The control protocol must support a number of data storage media types. Only the minimum set of SCSI commands needed to download mission data from a memory cartridge are defined as “required.” FC-PLDA SCSI commands, features, and parameters not defined as “required” for this standard are redefined as “allowed” and may be implemented as appropriate. Table [10-1](#) provides the four “required” SCSI commands and their features and parameter usage definitions.

---

<sup>1</sup> Document published as ANSI INCITS TR19-1998.

**TABLE 10-1.  
“REQUIRED” SCSI COMMANDS, FEATURES, AND PARAMETERS<sup>2</sup>**

COMMANDS/FEATURES	INITIATOR	TARGET	NOTES
<b>INQUIRY</b>	I	R	
Standard INQUIRY data (bytes 0-35)	I	R	
EVPD = 1	I	R	
Vital Product Data page codes:			
hex'00' (supported vital product pages)	I	R	
hex'80' (unit serial number page)	I	R	
hex'81' (implemented operations definition pg.)	I	A	
hex'82' (ASCII implemented operations def. pg.)	I	A	
hex'83' (device identification page)	I	R	
<b>READ (10)</b>	I	R	
DPO = 0	I	A	1
DPO = 1	I	A	1
FUA = 0	I	A	2
FUA = 1	I	A	2
RelAdr = 0	R	R	
RelAdr = 1	P	P	3
<b>READ CAPACITY</b>	I	R	
RelAdr = 0	R	R	
RelAdr = 1	P	P	3
PMI = 0	I	R	
PMI = 1	I	A	
<b>TEST UNIT READY</b>	I	R	
<p><b>Notes:</b></p> <ol style="list-style-type: none"> <li>1. The Disable Page Out (DPO) bit is associated with a device data caching policy.</li> <li>2. The Force Unit Access (FUA) bit is associated with whether the device may or may not return the requested Read data from its local cache.</li> <li>3. Relative Offset is prohibited since this requires the use of linking that is prohibited.</li> </ol> <p>P = Prohibited: The feature shall not be used between FC-PLDA compliant devices.  R = Required: The feature or parameter value shall be implemented by FC-PLDA compliant devices.  A = Allowed: The feature or parameter may be used between FC-PLDA compliant devices. The initiator determines if an “allowed” feature/parameter is supported via a required discovery process or a minimal response by the recipient.  I = Invokable: The feature or parameter may be used between FC-PLDA compliant devices. The recipient shall support “invokable” features or provide a response that it is not implemented as defined by the appropriate standard.</p>			

<sup>2</sup> Adapted from STANAG 4575, Table B-1.

## 10.5 Interface File Structure Definition

This interface file structure definition is adopted from STANAG 4575.<sup>3</sup> This file structure was selected to facilitate host computing platform independence and commonality. By incorporating an independent file structure backward and forward compatibility is ensured for the life of the standard.



This file structure definition does not define how data is physically stored on the recorder media, but provides a standardized method for access of the stored data at the interface. Data can be physically organized any way appropriate to the media, including multiple directories, as long as the file structure IAW section [10.5](#) is maintained or seen at the interface (section [10.4](#)).

10.5.1 Data Organization. A data recording can contain a single file, which is composed of one or more types of packetized data, or multiple files, in which one or more types of data are recorded simultaneously in separate files. For a recording file to be in compliance with this standard, it must contain, as a minimum, the following:

- a. Computer Generated Packet, Setup Record Format 1, in accordance with (IAW) paragraph [10.6.7.2.2](#) as the first packet in the recording.
- b. Time Data Packet(s) IAW paragraph [10.6.3](#) as the first dynamic packet after the Computer Generated Packet Set Record.
- c. One or more data format packets IAW section [10.6](#).

Multiple recordings may reside on the media, and each recording may contain one or more compliant files.

10.5.1.1 Data Hierarchy: The structures used to define the data stored according to this standard shall have the following relationships (highest to lowest), as depicted in Figure [10-2](#).

10.5.1.1.1 Directory: One or more directory blocks of data comprising a list of all Data Files located under the guidance of this Standard. Also contains supporting data that may be of interest to those manipulating the Data Files. The list of files is made up from "File Entries." The Directory shall always start at logical address zero of each directory block.

10.5.1.1.2 Directory Block: A block containing directory entries and other metadata.

---

<sup>3</sup> Annex B Protocol Interface Definitions, section 3, File Structure Definition.



10.5.1.1.3 Directory Block File Entry: A fixed-length data structure used to describe files. It contains the name, the starting address, the number of blocks of data assigned to the Data File, the total number of bytes contained in the file, and the file's creation date and time. It also contains a reserved field for future growth and a vendor-unique field.

10.5.1.1.4 Data Files: Data files are comprised of user data, presented at the interface in monotonically increasing contiguous logical addresses per file. Thus, if a file starts at logical address X, the next location containing file data must be at the next logical address, X+1, and the next location after that must be at the next logical address, X+2, etc.

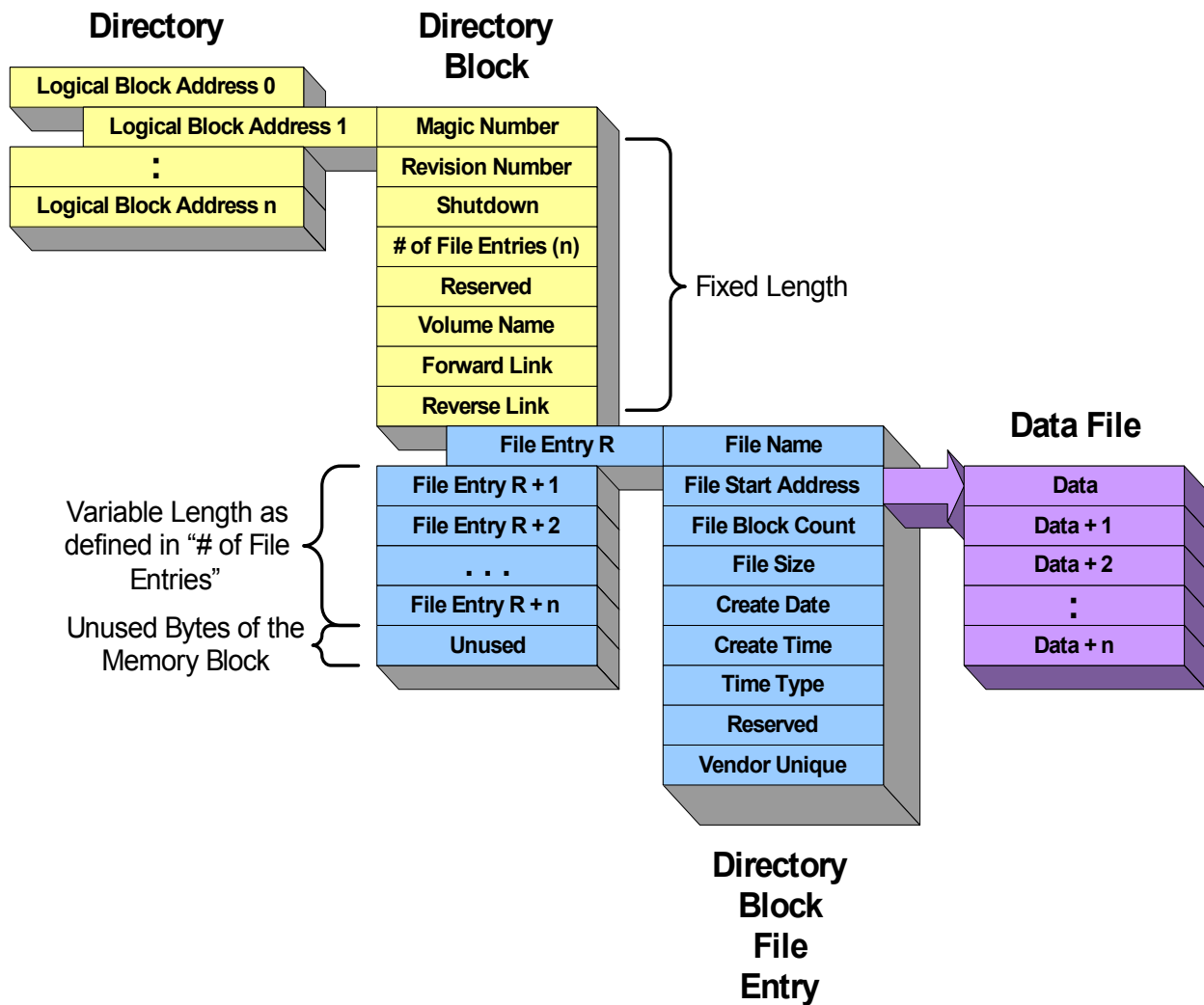


Figure 10-2. Directory Block structure.

10.5.2 Directory Definition: Name and location information for all files is recorded in a Directory (see Figure 10-2). The Directory is composed of one or more directory blocks as shown in Figure 10-3. At least one Directory Block is required and it must be located at SCSI logical block address 1. (Logical block address 0 is defined as a vendor unique area; its contents are not described or controlled by this standard.)

10.5.2.1 Directory Fixed Fields: The fixed fields within a Directory Block are used to name the volume of data, identify the number of entries, and to provide pointers to other addresses that contain additional directory blocks. The forward and backward link to the next address for the next Directory Block (if any) as well as back to the preceding Directory Block (if any). This allows for directory expansion beyond a single block and does not limit the placement of directory information.

10.5.2.2 Block Size: The media types used to implement this standard have varying block lengths. Some will have blocks as small as 512 bytes; others may have blocks as large as 64K bytes or larger. The block size used by a given media can be determined via the SCSI protocol and is not defined here.

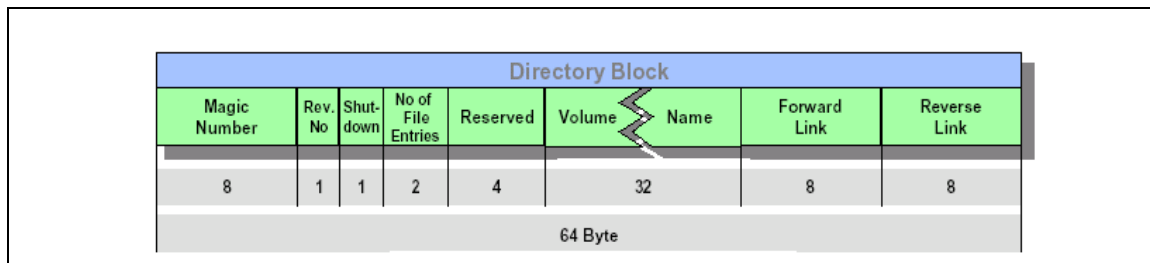


Figure 10-3. Directory Block (as depicted in STANAG 4575).

10.5.2.3 Directory to Data File Link: Each Data File on the media has a directory entry within a Directory Block that describes the file, as shown in Table 10-2. The directory entry for a data file contains a link to the starting location of the data contained in each file and the total number of blocks assigned for the storage of data, as shown in Table 10-3. This standard does not define the meaning of the data recorded within these Data File blocks.

<b>TABLE 10-2. DIRECTORY BLOCK FORMAT – PRIMARY BLOCK</b>			
<b>FIELD NAME</b>	<b>BYTES</b>	<b>DESCRIPTION</b>	<b>DATA TYPE</b>
Magic Number	8	An identifier for a directory block. The value is ASCII “FORTYtwo” (Hex – 0x464F52545974776F)	ASCII
Revision Number	1	Revision number of the standard compiled by the recording system. Example first version 00000001.	Unsigned Binary
Shutdown	1	Flag, if cleared to a 00h indicates that the volume was not properly dismounted, if seen on power-up is an indication that the directory chain may be faulty. If set = 0xFF, then file system properly shutdown. This field is only valid in the first directory block; other directory blocks set to 0xFF.	Unsigned Binary
Number of File Entries	2	Defines the number of file entries that follow in this block.	Unsigned Binary
Reserved	4	Fill with 0xFF	Unsigned Binary
VolName	32	Volume name. Fill with all 0x00 for no name.	ASCII
Forward Link	8	Block address of the next block containing directory information. Set equal to address of this block if this is the end of the chain.	Unsigned Binary
Reverse Link	8	Block address of the directory block pointing to this block. Set equal to the address of this block if this is the start of the chain.	Unsigned Binary
(n File Entries)	112 *n	One entry for each file. Total number of directory entries for this directory block as defined by field “Number of Directory Entries”.	See Table 10-3
Padding	Varies with n and Block Size	Pad to block boundary. Default value is 0xFF	Unsigned Binary
<b>Note:</b> 64 bytes in fixed fields.			

**TABLE 10-3. DIRECTORY ENTRY FORMAT**

<b>FIELD NAME</b>	<b>BYTES</b>	<b>DESCRIPTION</b>	<b>DATA TYPE</b>
Name	56	File name, see character set for restrictions. Fill with FFh for unused directory entries.	ASCII
FileStartAdd	8	Zero based address of the first block reserved for data associated with this file. Fill with FFh for unused directory entries.	Unsigned Binary
FileBlkCnt	8	One-based number that is the count of consecutive address blocks reserved for data for this file including the block pointed to by the FileStartAdd field. Fill with 00h for unused directory entries.	Unsigned Binary
FileSize	8	The actual number of bytes contained in this file. This file size will be equal to or less than the FileBlkCnt multiplied by the blocksize.	Unsigned Binary
File Create Date	8	DDMMYYYY ASCII character values, with no embedded spaces or other formatting characters, representing the numeric date on which the file was created (e.g. ASCII codes for the decimal digits 02092000 → 30h 32h 30h 39h 32h 30h 30h 30h represents 2 September 2000). Data for this is optional and shall be filled with 30h if this value is not available.	ASCII
File Create Time	8	HHMMSSss character values, with no embedded spaces or other formatting characters, representing the numeric time at which the file was created. HH is the number of the 24 hour based hour, MM is the number of minutes after the hour, SS is the number of seconds after the minute, and ss is the hundredths of seconds after the second.  Data for this is optional and shall be filled with 30h if this value is not available. Fill with 20h if this entire field is not available. Fill the portions of the field with 30h if a portion of the field, e.g., “ss” is not available.	ASCII
Time Type	1	A numeric code that qualifies the time and date values recorded in the “Create Date” and “Create Time” and “Close Time” fields. 00h = Coordinated Universal Time (Zulu) 01h = Local Time 02h – FF = TBD	Unsigned Binary
Reserved	7	Bytes in this region are reserved for future growth. Fill with 00h.	Unsigned Binary

<b>TABLE 10-3 (Cont'd). DIRECTORY ENTRY FORMAT</b>			
<b>FIELD NAME</b>	<b>BYTES</b>	<b>DESCRIPTION</b>	<b>DATA TYPE</b>
File Close Time	8	HHMMSSss character values, with no embedded spaces or other formatting characters, representing the numeric time at which the file was closed. HH is the number of the 24 hour based hour, MM is the number of minutes after the hour, SS is the number of seconds after the minute, and ss is the hundredths of seconds after the second.  Data for this is optional and shall be filled with 30h if this value is not available. Fill with 20h if this entire field is not available. Fill the portions of the field with 30h if a portion of the field, e.g., "ss" is not available.	ASCII
<b>Note:</b> 112 bytes in fixed fields.			

10.5.2.4 File Entry Name. Each file entry in a directory shall have a unique name (see character set paragraphs [10.5.3.2.1](#) and [10.5.3.2.2](#)).

10.5.2.5 Entry Singularity. Multiple Directory entries are not permitted to refer to the same data, either partially or completely.

10.5.2.6 Directory Entries and Fields. Directory block fields and entries shall be logically contiguous.

### 10.5.3 Data Definitions

10.5.3.1 Byte Order: The structures described in this Standard are defined to have the following bit and byte orientation. The least significant byte shall be transmitted first, the least significant bit of each byte shall be transmitted first, and data is read from the lowest logical address first. Fields defined with data type of ASCII are stored in the order shown in the field description, starting with the lowest logical address. Directory blocks are stored in the order shown in the tables, starting with the lowest logical address.

Assuming a 32-bit entry, composed of four 8-bit bytes, where the first and least significant byte is byte [0] and the last and most significant is byte [3], then the correspondence of bits to bytes, where bit [00] is the least significant bit, is as follows in Table [10-4](#).

TABLE 10-4. CORRESPONDENCE OF BITS TO BYTES								
Bit:	7	6	5	4	3	2	1	0
Byte [3]	31	30	29	28	27	26	25	24
Byte [2]	23	22	21	20	19	18	17	16
Byte [1]	15	14	13	12	11	10	9	8
Byte [0]	7	6	5	4	3	2	1	0

10.5.3.2 Naming Restrictions. The following rules shall be applied when forming names in order to assure the highest degree of interchange among other operating systems.

10.5.3.2.1 Characters. Characters from the first 127 common ASCII characters (00h through 7Eh) may be used in names except for specific prohibited characters.

Any ASCII character code value smaller than 20h is prohibited, except where the <00h> is used to terminate the name. The other prohibited characters with their hexadecimal representation are defined in Table [10-5](#).

TABLE 10-5. PROHIBITED CHARACTERS HEXIDECIMAL REPRESENTATION			
Forbidden Characters in Names	Hexadecimal Value	Forbidden Characters in Names	Hexadecimal Value
”	22h	=	3Dh
‘	27h	>	3Eh
*	2Ah	?	3Fh
/	2Fh	\	5Ch
:	3Ah	]	5Dh
;	3Bh	[	5Eh
<	3Ch		7Ch

10.5.3.2.2 Names. Names used for this interface will observe the following rules:

- a. Filenames shall not be case sensitive.
- b. Leading and trailing spaces are not permitted.
- c. Leading periods are not permitted.
- d. Names shall be left-justified in the field and are terminated with a null<00h>. The maximum length of the name is the length of the field minus one.

## 10.6 Data Format Definition

10.6.1 Common Packet Elements. Data shall have three required parts, a Packet Header, a Packet Body, a Packet Trailer, and an optional part if enabled, a Packet Secondary Header. See Figure 10-4 for a diagram of the generic packet format. This figure does not depict the bit lengths of each field. Depending on the data type, 8-bit, 16-bit and 32-bit word sizes are used. The size of a single packet may be a maximum of 524 288 bytes, with one exception. The first packet in the file must be a Computer Generated Data Packet, Format 1 Setup Records, may be a maximum of 134 217 728 bytes.

With the exception of Computer Generated Packets and Time Data Packets, all other packets shall be generated within 100 milliseconds whenever data is available. This requirement ensures that a packet shall contain less than 100 milliseconds worth of data, and that a packet containing any data must be generated within 100 milliseconds from the time the first data was placed in the packet. This strategy will assure packet granularity but save bandwidth by not forcing or marking empty/idle packets.

A packet has the basic structure shown in Figure [10-4](#). Note that the width of the structure is not related to any number of bits. This figure merely represents the relative packet elements and their placement within the packet.

<b>PACKET SYNC PATTERN</b>	<b>Packet Header</b>
<b>CHANNEL ID</b>	
<b>PACKET LENGTH</b>	
<b>DATA LENGTH</b>	
<b>HEADER VERSION</b>	
<b>SEQUENCE NUMBER</b>	
<b>PACKET FLAGS</b>	
<b>DATA TYPE</b>	
<b>RELATIVE TIME COUNTER</b>	
<b>HEADER CHECKSUM</b>	
<b>TIME</b>	
<b>RESERVED</b>	<b>Secondary Header (Optional)</b>
<b>SECONDARY HEADER CHECKSUM</b>	
<b>CHANNEL SPECIFIC DATA</b>	<b>Packet Body</b>
<b>INTRA-PACKET TIME STAMP 1</b>	
<b>INTRA-PACKET DATA HEADER 1</b>	
<b>DATA 1</b>	
<b>⋮</b>	
<b>INTRA-PACKET TIME STAMP n</b>	
<b>INTRA-PACKET DATA HEADER n</b>	
<b>DATA n</b>	
<b>DATA CHECKSUM</b>	<b>Packet Trailer</b>

Figure 10-4. General Packet format.



To further clarify the packet layout, Figure 10-5 shows the generic packet in a 32-bit, little-endian format, and assumes 16-bit data words and data checksum.

<b>msb</b> <b>31</b>	<b>16</b>	<b>15</b>	<b>lsb</b> <b>0</b>	
<b>CHANNEL ID</b>		<b>PACKET SYNC PATTERN</b>		<b>Packet Header</b>
<b>PACKET LENGTH</b>				
<b>DATA LENGTH</b>				
<b>DATA TYPE</b>	<b>PACKET FLAGS</b>	<b>SEQUENCE NUMBER</b>	<b>HEADER VERSION</b>	
<b>RELATIVE TIME COUNTER</b>				
<b>HEADER CHECKSUM</b>		<b>RELATIVE TIME COUNTER</b>		
<b>TIME (LSLW)</b>				
<b>TIME (MSLW)</b>				<b>(Optional) Packet Secondary Header</b>
<b>SECONDARY HEADER CHECKSUM</b>		<b>RESERVED</b>		
<b>CHANNEL SPECIFIC DATA</b>				<b>Packet Body</b>
<b>INTRA-PACKET TIME STAMP 1</b>				
<b>INTRA-PACKET TIME STAMP 1</b>				
<b>INTRA-PACKET DATA HEADER 1</b>				
<b>DATA 1 WORD 2</b>		<b>DATA 1 WORD 1</b>		
<b>DATA 1 WORD n</b>		<b>:</b>		
<b>INTRA-PACKET TIME STAMP 2</b>				
<b>INTRA-PACKET TIME STAMP 2</b>				
<b>INTRA-PACKET DATA HEADER 2</b>				
<b>DATA 2 WORD 2</b>		<b>DATA 2 WORD 1</b>		
<b>DATA 2 WORD n</b>		<b>:</b>		
<b>:</b>				
<b>INTRA-PACKET TIME STAMP N</b>				
<b>INTRA-PACKET TIME STAMP N</b>				
<b>INTRA-PACKET DATA HEADER N</b>				
<b>DATA N WORD 2</b>		<b>DATA N WORD 1</b>		
<b>DATA N WORD n</b>		<b>:</b>		
<b>[FILLER]</b>				
<b>DATA CHECKSUM</b>				<b>Packet Trailer</b>

Figure 10-5. A 32-bit Packet format layout.

Depending on the data type, the size of the Data Checksum can be 16-bits, 32-bits, 8-bits, or left out entirely. For a 32-bit Data Checksum, the packet trailer would be as shown in Figure 10-6:

<b>msb</b> 7	<b>lsb</b> 0	
[ FILLER ]		Packet Trailer
DATA CHECKSUM (LSB)		
----- DATA CHECKSUM		
----- DATA CHECKSUM		
----- DATA CHECKSUM (MSB)		

Figure 10-6. Packet trailer for 32-bit Data Checksum.

For an 8-bit Data Checksum, the packet trailer would be as shown in Figure 10-7:

<b>msb</b> 7	<b>lsb</b> 0	
[ FILLER ]		Packet Trailer
DATA CHECKSUM (LSB)		

Figure 10-7. Packet trailer for 8-bit Data Checksum.

10.6.1.1 Packet Header. The length of the Packet Header is fixed at 24 bytes (192-bits). The Packet Header is mandatory and shall consist of the ten fields, positioned contiguously in the following sequence:

10.6.1.1.1 Packet Sync Pattern: This field (2 bytes) contains a static sync value for every packet. The Packet Sync Pattern value shall be 0xEB25.

10.6.1.1.2 Channel ID. This field (2 bytes) contains a value representing the Packet Channel ID. All channels in a system must have a unique value. Channel value 0x0000 is reserved and is used to insert computer-generated messages into the composite data stream. Channel values 0x0001 thru 0xFFFF are available.

10.6.1.1.3 Packet Length. This field (4 bytes) contains a value representing the length of the entire packet. The value shall be in bytes and is always a multiple of four (bits 1 and 0 shall

always be zero). This Packet Length includes the Packet Header, Packet Secondary Header (if enabled), Channel Specific Data, Intra-Packet Data Headers, Data, Filler, and Data Checksum.

10.6.1.1.4 Data Length. This field (4 bytes) contains a value representing the valid Data Length within the packet. This value shall be represented in bytes. Valid data length includes Channel Specific Data and Intra-Packet Data Headers but does not include Filler.

10.6.1.1.5 Header Version. This field (1 byte) contains a value representing the version of the header. The value shall be represented by the following bit patterns:

0x00 = Reserved.

0x01 = Initial Release Header Version.

0x02 thru 0xFF = Reserved for future releases.

10.6.1.1.6 Sequence Number. This field (1 byte) contains a value representing the Packet Sequence Number. This is simply a counter that marks in increments from 0x00 to 0xFF for every packet transferred from a particular channel.



Sequence number counter will repeat if more than 256 packets are transferred in a given recording per channel.

10.6.1.1.7 Packet Flags. This field (1 byte) contains bits representing information on the content and format of the packet(s) as follows:

Bit 7: Indicates the presence or absence of the Packet Secondary Header.

0 = Packet Secondary Header is not present.

1 = Packet Secondary Header is present.

Bit 6: Indicates the Intra-Packet Time Stamp Type.

0 = Packet Header 48-bit Relative Time Counter.

1 = Same as Packet Secondary Header Time (bit 7 must = 1).

Bit 5: Indicates the Relative Time Counter Sync Error.

0 = No Relative Time Counter Sync Error.

1 = Relative Time Counter Sync Error has occurred.

Bit 4: Indicates the Data Overflow Error.

0 = No data overflow.

1 = Data overflow has occurred.

Bits 3-2: Indicates the Packet Secondary Header Time Format.

00 = IRIG 106 Chapter 4 binary weighted 48-bit time format. The two LSBs of the 64-bit Packet Secondary Header Time and Intra-Packet Data Header time shall be zero filled.

01 = Reserved.

10 = Reserved.

11 = Reserved.

Bits 1-0: Indicates Data Checksum existence.

00 = No Data Checksum present.

01 = 8-bit Data Checksum present.

10 = 16-bit Data Checksum present.

11 = 32-bit Data Checksum present.

10.6.1.1.8 Data Type. This frame (1 byte) contains a value representing the type and format of the data. All values not used to define a data type are reserved for future data type growth:

0x00 = Computer Generated Data: Format 0 - (user defined)

0x01 = Computer Generated Data: Format 1 - (setup record)

0x09 = PCM Data: Format 1

0x11 = Time Data: Format 1

0x19 = MIL-STD-1553 Data: Format 1

0x21 = Analog Data: Format 1

0x29 = Discrete Data: Format 1

0x30 = Message Data: Format 0

0x38 = ARINC 429 Data: Format 0

0x40 = Video Data: Format 0 - (MPEG-2 Video)

0x48 = Image Data: Format 0 - (Image)

0x50 = UART Data: Format 0

10.6.1.1.9 Relative Time Counter. This frame (6 bytes) contains a value representing the Relative Time Counter.



This is a free-running 10 MHz binary counter represented by 48 bits common to all data channels. The counter shall be derived from an internal crystal oscillator and shall remain free running during each recording session. The applicable data bit to which the 48-bit value applies will be defined in each data type section.

10.6.1.1.10 Header Checksum. This field (2 bytes) contains a value representing a 16-bit arithmetic sum of all header bytes excluding the Header Checksum Word.

10.6.1.2 Packet Secondary Header (optional). The length of the Packet Secondary Header is fixed at 12 bytes (96 bits). The Packet Secondary Header is optional and, when enabled, shall consist of the following three fields positioned contiguously in the following sequence:

10.6.1.2.1 Time. This field (8 bytes) contains the value representing Time in the format indicated by bits 2 and 3 of the Packet Flags as specified in paragraph 10.6.1.1.7.

10.6.1.2.2 Reserved. This field (2 bytes) is reserved and shall be zero filled.

10.6.1.2.3 Secondary Header Checksum. This field (2 bytes) contains a value representing a 16-bit arithmetic sum of all Secondary Header bytes, but excluding the Secondary Header Checksum Word.

10.6.1.3 Packet Body. The format of the data in the packet body is unique to each channel type. Detailed descriptions of the type-specific data formats found in packet bodies are described in subsequent sections of this document.

10.6.1.3.1 Channel Specific Data. This field (variable bytes) contains the number and contents of the Channel Specific Data field(s) depending on the Data Type field in the Packet Header. Channel Specific Data is mandatory for each data type and channel. The occurrence of Channel Specific Data is once per packet and precedes packet channel data.

10.6.1.3.2 Intra-Packet Time Stamp. This field (8 bytes) contains Time in either 48-bit relative format (plus 16 high-order zero bits) or 64-bit absolute format, as specified in the Packet Flags in the Packet Header. The intra-packet time stamps are only mandatory where defined by the data formats.

10.6.1.3.3 Intra-Packet Data Header. This field (variable bytes) contains additional status and format information pertaining to the data items that follow. The intra-packet data headers are only mandatory where defined by the data formats.



The Intra-Packet Time Stamp and the Intra-Packet Data Header are collectively called the Intra-Packet Header. In some cases an Intra-Packet Header may only have a time stamp (zero-length data header), while in other cases, the Intra-Packet Header only has a data header (zero-length time stamp). Some data types have no Intra-Packet Header. The Intra-Packet Header requirements are specified separately for each data type.

10.6.1.3.4 Data. This field (n bytes) contains valid data from a particular channel as defined within the data formats contained within this standard.

10.6.1.4 Packet Trailer. The packet trailer may contain Filler, a Data Checksum, both Filler and a Data Checksum, or neither Filler nor a Data Checksum. In the latter case, the packet trailer has zero length. The reason a packet trailer would have a zero length is best explained by understanding the reason for inserting Filler. The purpose of the Filler is to:

keep all packets aligned on 32-bit boundaries (i.e., make all packet lengths a multiple of 4 bytes), and

optionally, keep all packets from a particular channel the same length.

If both these requirements are already met without adding Filler, then Filler will not be added.

The inclusion of the Data Checksum is optional as well and is indicated by the Packet Flags setting. When included, the packet trailer contains either an 8-bit, 16-bit or 32-bit Data Checksum. Depending on the Packet Flags option selected, the Data Checksum is the arithmetic sum of all of the bytes (8-bits), words (16-bits) or long words (32-bits) in the packet, excluding the 24 bytes of Packet Header Words, Packet Secondary Header (if enabled) and the Data Checksum word. Stated another way, the Data Checksum includes everything in the packet body plus all added Filler.

10.6.1.4.1 Filler. This field (n bytes/bits) contains Filler to make the packet size a multiple of 4 bytes and (optionally) make all packets from a channel the same size.

10.6.1.4.2 8-Bit Data Checksum. This field (1 byte) contains a value representing an 8-bit arithmetic sum of the bytes in the packet (includes Channel Specific Data, Data, and Filler). Only inserted if Packet Flag bits 0 and 1 = 01.

10.6.1.4.3 16-Bit Data Checksum. This field (2 bytes) contains a value representing a 16-bit arithmetic sum of the words in the packet (includes Channel Specific Data, Data, and Filler) and is only inserted if Packet Flag bits 0 and 1 = 10.

10.6.1.4.1 32-Bit Data Checksum. This field (4 bytes) contains a value representing a 32-bit arithmetic sum of the long words in the packet (includes Channel Specific Data, Data, and Filler). Only inserted if Packet Flag bits 0 and 1 = 11.

10.6.2 PCM Data Packets, Format 1. A packet with PCM data has the basic structure shown in Figure [10-8](#). Note that the width of the structure is not related to any number of bits. This figure merely represents the relative placement of data in the packet.

PACKET HEADER
CHANNEL SPECIFIC DATA
<i>(Optional)</i> INTRA-PACKET TIME STAMP
<i>(Optional)</i> INTRA-PACKET DATA HEADER
MINOR FRAME DATA
<i>(Optional)</i> INTRA-PACKET TIME STAMP
<i>(Optional)</i> INTRA-PACKET DATA HEADER
MINOR FRAME DATA
<i>(Optional)</i> INTRA-PACKET TIME STAMP
<i>(Optional)</i> INTRA-PACKET DATA HEADER
MINOR FRAME DATA
<i>(Optional)</i> INTRA-PACKET TIME STAMP
<i>(Optional)</i> INTRA-PACKET DATA HEADER
MINOR FRAME DATA
<i>(Optional)</i> INTRA-PACKET TIME STAMP
<i>(Optional)</i> INTRA-PACKET DATA HEADER
MINOR FRAME DATA
:
<i>(Optional)</i> INTRA-PACKET TIME STAMP
<i>(Optional)</i> INTRA-PACKET DATA HEADER
MINOR FRAME DATA
PACKET TRAILER

Figure 10-8. General PCM Data Packet, Format 1.

The user may separately enable or disable word unpacking on each active PCM channel. Word unpacking will force the least significant bit of each word to be aligned on a 16-bit boundary. High-order filler bits are added to words, as necessary, to force alignment.

The user may separately enable or disable frame synchronizing on each active PCM channel. This provides a Throughput Mode that will transfer data to the packet without frame synchronization. Throughput Mode essentially disables all setup and packing/unpacking options for the packet, and just puts data in the packet as it is received.

10.6.2.1 PCM Packet Channel Specific Data. The packet body portion of each PCM packet begins with the Channel Specific Data, as shown in Figure [10-9](#).

<b>msb</b>								<b>lsb</b>
31	30	29	28	27	24	23	18	17
R	IPH	MA	MI	LOCKST	MODE	SYNCOFFSET		
								0

Figure 10-9. PCM Packet Channel Specific Data format.

where:

Bits 17-0: Sync Offset (SYNCOFFSET): field contains an 18-bit binary value representing the Word offset into the major frame for the first data word in the packet. Not valid for Packed or Throughput Mode.

Bits 23-18: Mode (MODE): indicates the data packing mode.

Bits 23-21 are reserved.

Bit 20 indicates Throughput Data Mode.

0 = Throughput Data Mode not enabled.

1 = Throughput Data Mode enabled.

Bit 19 indicates Packed Data Mode.

0 = Packed Data Mode not enabled.

1 = Packed Data Mode enabled.

Bit 18 indicates Unpacked Data Mode.

0 = Unpacked Data Mode not enabled.

1 = Unpacked Data Mode enabled.

Bits 27-24: Lock status (LOCKST): indicates the lock status of the frame synchronizer. Not valid for Throughput Mode.

Bits 27-26 indicate Minor Frame Status.

00 = Reserved.

01 = Reserved.

10 = Minor Frame Check (after losing Lock).

11 = Minor Frame Lock.

Bits 25-24 indicate Major Frame Status.

00 = Minor Frames only.

01 = Reserved.

10 = Major Frame Check (after losing Lock).

11 = Major Frame Lock.



- Bit 28: Minor Frame Indicator (MI) indicates if the first word in the packet is the beginning of a minor frame. Not valid for throughput mode.
- 0 = First word is not the beginning of a minor frame.  
1 = First word is the beginning of a minor frame.
- Bit 29: Major Frame Indicator (MA): indicates if the first word in the packet is the beginning of a major frame. Not valid for Throughput Mode.
- 0 = First word is not the beginning of a major frame.  
1 = First word is the beginning of a major frame.
- Bit 30: Intra-Packet Header (IPH): indicate if Intra-Packet Headers (Intra-Packet Time Stamp and Intra-Packet Data Header) are inserted before each minor frame. Intra-Packet Headers are only optional because of the mode selection. This determines whether Intra-Packet Headers are included or omitted.
- 0 = Intra-Packet Headers are omitted for Throughput Mode.  
1 = Intra-Packet Headers are required for Packed Data and Unpacked Data modes.
- Bit 31: Reserved.

10.6.2.2 PCM Packet Body. After the Channel Specific Data, the PCM data and Intra-Packet Headers are inserted in the packet in integral numbers of minor or major frames, unless the packet is in Throughput Mode. In Throughput Mode, there is no frame or word alignment to the packet data and no Intra-Packet Headers are inserted in the data.

10.6.2.2.1 PCM Data in Unpacked Mode. In Unpacked Mode, packing is disabled and each data word is padded with the number of Filler bits necessary to align the first bit of each word with the next 16-bit boundary in the packet. For example, 4 pad bits are added to 12 bit words, 6 pad bits are added to 10 bit words, etc.

Minor frame sync patterns larger than 16 bits are divided into two words of packet data. If the sync pattern has an even number of bits, then it will be divided in half and placed in two packet words. For example, a 24-bit sync pattern is broken into two 12-bit words with 4 bits of pad in each word. If the sync pattern has an odd number of bits, it is broken into two words with the second word having one-bit more of the sync pattern. For example, if the minor sync pattern is 25 bits, then the first sync word is 12 bits of sync pattern plus 4 bits of pad, and the second sync word is 13 bits of sync pattern plus 3 bits of pad.

Given PCM frames with a 24-bit minor sync pattern and n data words where the bit lengths of data words 1, 2, and 3 are 12, 16, and 8, respectively. The resultant PCM packets are as shown in Figure [10-10](#):

<b>msb</b> <b>15</b>	<b>lsb</b> <b>0</b>
PACKET HEADER	
CHANNEL SPECIFIC DATA (BITS 15-0)	
CHANNEL SPECIFIC DATA (BITS 31-16)	
INTRA-PACKET TIME STAMP (BITS 15-0)	
INTRA-PACKET TIME STAMP (BITS 31-16)	
INTRA-PACKET TIME STAMP (BITS 47-32)	
INTRA-PACKET TIME STAMP (BITS 63-48)	
INTRA-PACKET DATA HEADER (BITS 15-0)	
4-BITS PAD	12-BITS SYNC (BITS 23-12)
4-BITS PAD	12-BITS SYNC (BITS 11-0)
4-BITS PAD	12-BITS WORD 1 DATA
16-BITS WORD 2 DATA	
8-BITS PAD	8-BITS WORD 3 DATA
:	
WORD <i>n</i> DATA BITS + PAD IF NEEDED	
INTRA-PACKET TIME STAMP ( BITS 15-0)	
INTRA-PACKET TIME STAMP ( BITS 31-16)	
INTRA-PACKET TIME STAMP ( BITS 47-32)	
INTRA-PACKET TIME STAMP ( BITS 63-48)	
INTRA-PACKET DATA HEADER ( BITS 15-0)	
:	
REPEAT FOR EACH MINOR FRAME	
:	
PACKET TRAILER	

Figure 10-10. PCM Data – Unpacked Mode sample packet.

10.6.2.2.2 PCM Data in Packed Mode. In Packed Mode, packing is enabled and pad is not added to each data word. However, if the number of bits in the minor frame is not an integer multiple of 16, then ‘Y’ Filler bits will be added to the end of each minor frame of bit length L.  $Y = 16 - \text{MOD}(L,16)$ , or 16 minus the integer remainder when L is divided by 16. In packed mode, the PCM stream is minor frame synchronized so the first data bit in the packet is the first data bit of a minor frame. If  $X = 16 - Y$ , then the resultant PCM packets are as shown in Figure 10-11:

<b>msb</b> <b>15</b>	<b>lsb</b> <b>0</b>
PACKET HEADER	
CHANNEL SPECIFIC DATA (BITS 15-0)	
CHANNEL SPECIFIC DATA (BITS 31-16)	
INTRA-PACKET TIME STAMP ( BITS 15-0)	
INTRA-PACKET TIME STAMP ( BITS 31-16)	
INTRA-PACKET TIME STAMP ( BITS 47-32)	
INTRA-PACKET TIME STAMP ( BITS 63-48)	
INTRA-PACKET DATA HEADER ( BITS 15-0)	
DATA BITS 0 to 16	
DATA BITS 16 to 31	
DATA BITS 32 to 47	
:	
Y FILLER BITS	X DATA BITS
INTRA-PACKET TIME STAMP ( BITS 15-0)	
INTRA-PACKET TIME STAMP ( BITS 31-16)	
INTRA-PACKET TIME STAMP ( BITS 47-32)	
INTRA-PACKET TIME STAMP ( BITS 63-48)	
INTRA-PACKET DATA HEADER ( BITS 15-0)	
:	
REPEAT FOR EACH MINOR FRAME	
:	
PACKET TRAILER	

Figure 10-11. PCM Data – Packed Mode Sample Packet.

10.6.2.2.3 PCM Data in Throughput Mode. In Throughput mode, the PCM data are not frame synchronized so the first data bit in the packet can be any bit in the major frame. The resultant PCM packets are as shown in Figure 10-12. Only bit 20 of the Channel Specific Data word is set to one, indicating Throughput Mode. All other bits of the Channel Specific Data word are undefined and shall be set to zero.

<b>msb</b> <b>15</b>	<b>lsb</b> <b>0</b>
PACKET HEADER	
CHANNEL SPECIFIC DATA (BITS 15-0)	
CHANNEL SPECIFIC DATA (BITS 31-16)	
DATA BITS 0 to 15	
DATA BITS 16 to 31	
DATA BITS 32 to 47	
:	
PACKET TRAILER	

Figure 10-12. PCM Data – Throughput Mode Sample Packet.

10.6.2.2.4 PCM Intra-Packet Header. When recording in Packed or Unpacked Mode, all PCM minor frames shall include an Intra-Packet Header containing a 64-bit Intra-Packet Time Stamp and a 16-bit Intra-Packet Data Header, which is inserted immediately before the minor frame sync pattern. The length of the Intra-Packet Header is fixed at 10 bytes (80-bits) positioned contiguously, in the following sequence (see Figure 10-13):

<b>msb</b> 31	15	12	11	<b>lsb</b> 0
TIME (LSLW)				
TIME (MSLW)				
			LOCKST	RESERVED

Figure 10-13. PCM Intra-Packet Header.

10.6.2.2.4.1 Intra-Packet Time Stamp. This field (8 bytes) indicates the time tag of the PCM minor frame. It is not valid for Throughput Mode. The first long word bits 31-0 and second long word bits 31-0 indicate the following values:

- The 48-bit Relative Time Counter that corresponds to the first data bit of the minor frame with bits 31 to 16 in the second long word zero filled; or

- Packet Secondary Header Time Type, if enabled by bit 6 in the Packet Header Flags (paragraph 10.6.1.1.7), corresponds to the time format indicated by bits 2 and 3 in the Packet Secondary Header Time (paragraph [10.6.1.1.7](#)) and to the first data bit of the minor frame.

#### 10.6.2.2.4.2 Intra-Packet Data Header

Bits 11-0: Reserved.

Bits 15-12: Lock status (LOCKST): indicates the Lock Status of the frame synchronizer for each minor frame.

Bits 15-14 indicate Minor Frame Status.

00 = Reserved.

01 = Reserved.

10 = Minor Frame Check (after losing Lock).

11 = Minor Frame Lock.

Bits 13-12 indicate Major Frame Status.

00 = Minor Frames only.

01 = Reserved.

10 = Major Frame Check (after losing Lock).

11 = Major Frame Lock.

10.6.3 Time Data Packets, Format 1. Time is treated like another data channel. A single Time Data Packet is inserted into the multiplexed data stream at least with the rate equal to the time code frame rate or resolution of the time type format. The 48-bit Relative Time Counter shall be captured for insertion into the Time Data Packet Header per the following:

10.6.3.1 IRIG Time Type Formats. At precisely the exact moment the IRIG “on-time” mark is received by the hardware indicating the start of a new IRIG time code frame, the 48-bit Relative Time Counter shall be captured for insertion into the Time Data Packet Header.

10.6.3.2 All Non-IRIG Time Type Formats. At precisely the exact moment of any 10-ms absolute time change received by the hardware, the 48-bit Relative Time Counter shall be captured for insertion into the Time Packet Data Header. After capture of the 48-bit Relative Time Counter, the decoded absolute time from the time code frame is inserted into the data portion of the Time Packet.



A Time Data Packet shall be the first dynamic data packet at the start of each recording. Only static Computer Generated Data packets may precede the Time Data Packet in the recording.

A packet with time data has the basic structure shown in Figure 10-14. Note that the width of the structure is not related to any number of bits. This drawing merely represents the relative placement of data in the packet. Time Packets do not have Intra-Packet Headers.

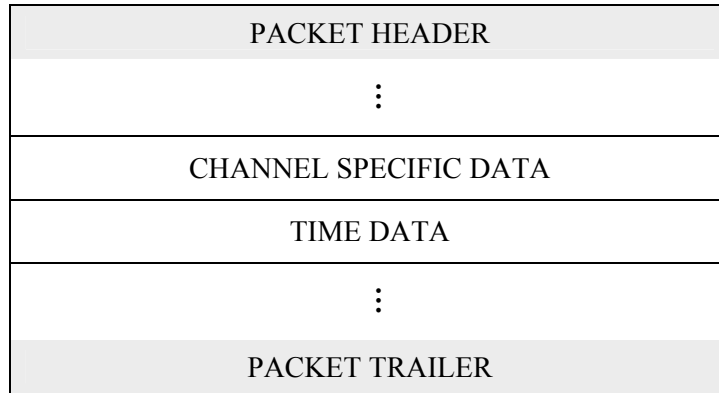


Figure 10-14. General Time Data Packet, Format 1.

10.6.3.3 Time Packet Channel Specific Data. The Packet Body portion of each Time Data Packet begins with a Channel Specific Data Word formatted as described in Figure 10-15.

<b>msb</b>									<b>lsb</b>
31		12	11	8	7	4	3		0
RESERVED			DATE		FMT		EXT		

Figure 10-15. Time Data Packet Channel Specific Data Word format.

where:

Bits 3-0: External Time (EXT): indicate if the external time source is present.

Bit 0 indicates if the external time source is present.

0 = External time is not present.

1 = External time present.

Bits 3-1 are reserved

Bits 7-4: Time Format (FMT): indicates the Time Data Packet format. All bit patterns not used to define a time format type are reserved for future data type growth.

0x0 = IRIG-B.

0x1 = IRIG-A.

0x2 = IRIG-G.

0x3 = Internal real-time clock.

0x4 = UTC time from GPS.

0x5 = Native GPS time.

0x6 thru 0xF = Reserved.

Bits 11-8: Date Format (DATE): indicates the Date Format. All bit patterns not used to define a date format type are reserved for future growth.

Bit 9 indicates Date Format.

0 = IRIG day available.

1 = Month and Year available.

Bit 8 indicates if this is a leap year.

0 = Not a leap year.

1 = Is a leap year.

Bits 11-10 are reserved.

Bits 31-12: Reserved.

10.6.3.4 Time Packet Body. After the Channel Specific Data word, the Time Data Words are inserted in the packet in binary coded decimal (BCD) format as shown in Figure 10-16.

<b>msb</b>										<b>lsb</b>	
<b>15</b>		12 11				8 7		4 3		<b>0</b>	
0	TSn			Sn			Hmn			Tmn	
0	0	THn			Hn			0	TMn		Mn
0	0	0	TOn			On			TDn		Dn
0	0	Oyn			HYn			TYn		Yn	
<b>LEGEND:</b>											
Tmn	Tens of milliseconds				TDn	Tens of days					
Hmn	Hundreds of milliseconds				HDn	Hundreds of days					
Sn	Units of seconds				On	Units of months					
TSn	Tens of seconds				TOn	Tens of months					
Mn	Units of minutes				Yn	Units of years					
TMn	Tens of minutes				TYn	Tens of years					
Hn	Units of hours				HYn	Hundreds of years					
THn	Tens of hours				OYn	Thousands of years					
Dn	Units of days				0	Always zero					

Figure 10-16. Time Data - Packet format, day, month, and year format.

10.6.4 MIL-STD-1553 Bus Data Packets, Format 1. MIL-STD-1553 BUS data is packetized in Message Mode, where each 1553 bus “transaction” is recorded as a “message.” A four-item Intra-Packet Data Header is inserted prior to each transaction. A transaction is a BC-to-RT, RT-to-BC, or RT-to-RT word sequence, starting with the command word and including all data and status words that are part of the transaction, or a mode code word broadcast. Multiple messages may be encoded into the data portion of a single packet.

10.6.4.1 MIL-STD-1553 Packet Channel Specific Data. The packet body portion of each MIL-STD-1553 Data Packet begins with a Channel Specific Data word formatted as described in Figure 10-17.

<b>msb</b>										<b>lsb</b>		
31 30 29		24 23										<b>0</b>
TTB		RESERVED				MSGCOUNT						

Figure 10-17. MIL-STD-1553 Packet Body Channel Specific Data Word format.



where:

- Bits 23-0: Message Count (MSGCOUNT): indicates the binary value of the number of messages included in the packet. An integral number of complete messages will be in each packet.
- Bits 29-24: Reserved.
- Bits 31-30: Time Tag Bits (TTB): indicates which bit of the MIL-STD-1553 message the Intra-Packet Header time tags.
  - 00 = Last bit of the last word of the message.
  - 01 = First bit of the first word of the message.
  - 10 = Last bit of the first (command) word of the message.
  - 11 = Reserved.

10.6.4.2 MIL-STD-1553 Packet Body. A packet with n-number of MIL-STD-1553 messages has the basic structure shown in Figure 10-18. Note that the width of the structure is not related to any number of bits. This figure merely represents the relative placement of data in the packet.

PACKET HEADER
CHANNEL SPECIFIC DATA
INTRA-PACKET TIME STAMP FOR MESSAGE 1
INTRA-PACKET DATA HEADER FOR MESSAGE 1
MESSAGE 1
INTRA-PACKET TIME STAMP FOR MESSAGE 2
INTRA-PACKET DATA HEADER FOR MESSAGE 2
MESSAGE 2
:
INTRA-PACKET TIME STAMP FOR MESSAGE n
INTRA-PACKET DATA HEADER FOR MESSAGE n
MESSAGE n
PACKET TRAILER

Figure 10-18. MIL-STD-1553 Data Packet, Format 1.

10.6.4.2.1 MIL-STD-1553 Intra-Packet Header. After the Channel Specific Data, the MIL-STD-1553 Data are inserted into the packet in messages. Each MIL-STD-1553 message is preceded by an Intra-Packet Header consisting of an Intra-Packet Time Stamp and an Intra-Packet Data Header.

10.6.4.2.1.1 MIL-STD-1553 Intra-Packet Time Stamp. This frame (8 bytes) indicates the time tag of the MIL-STD-1553 message as follows.

- The 48-bit Relative Time Counter that corresponds to the data bit indicated in the MIL-STD-1553 Channel Specific Data, Time Tag Bits (paragraph 10.6.4.1) with bits 31 to 16 in the second long word zero filled; or
- Packet Secondary Header Time Type, if enabled by bit 6 in the Packet Header Flags (paragraph 10.6.1.1.7), corresponds to the time format indicated by bits 2 and 3 in the Packet Secondary Header Time format (paragraph 10.6.1.1.7) and to the data bit indicated in the MIL-STD-1553 Channel Specific Data, Time Tag Bits (paragraph 10.6.4.1).

10.6.4.2.1.2 MIL-STD-1553 Intra-Packet Data Header. The length of the Intra-Packet Data Header is fixed at 6 bytes (48-bits) positioned contiguously in the following sequence as shown in Figure 10-19.

<b>msb</b>	<b>lsb</b>
15	0
BLOCK STATUS WORD	
GAP TIMES WORD	
LENGTH WORD	

Figure 10-19. MIL-STD-1553 Intra-Packet Data Header.

10.6.4.2.1.2.1 Block Status Word (BSW). Bits 15-0 contain the Block Status Word for both the message type and whether any 1553 bus protocol errors occurred during the message transfer. The Block Status Word bit definitions are as shown in Figure 10-20.

<b>msb</b>															<b>lsb</b>
15-14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	BID	ME	RR	FE	TM	R	R	R	LE	SE	WE	R	R	R	

Figure 10-20. Block Status Word bit definitions.


where:

- Bits 15-14: Reserved (R).
- Bit 13: Bus ID (BID): indicates the bus ID for the message.  
0 = Message was from Channel A.  
1 = Message was from Channel B.
- Bit 12: Message Error (ME): indicates a message error was encountered.  
0 = No message error.  
1 = Message error.
- Bit 11: RT to RT transfer (RR): indicates a RT to RT transfer; message begins with two command words.  
0 = No RT to RT transfer.  
1 = RT to RT transfer.
- Bit 10: Format Error (FE): indicates a frame error.  
0 = No format error.  
1 = Format error.
- Bit 9: Response Time Out (TM): indicates a response time out occurred.  
0 = No response time out.  
1 = Response time out.
- Bits 8-6: Reserved (R).
- Bit 5: Word Count Error (LE): indicates a word count error occurred.  
0 = No word count error.  
1 = Word count error.
- Bit 4: Sync Type Error (SE): indicates an incorrect sync type occurred.  
0 = No sync type error.  
1 = Sync type error.
- Bit 3: Invalid Word Error (WE): indicates an invalid word error occurred.  
0 = No invalid word error.  
1 = Invalid word error.
- Bits 2-0: Reserved (R).

10.6.4.2.1.2.2 Gap Times Word. The Gap Times Word indicates the number of tenths of microseconds in length of the internal gaps within a single transaction. For most messages, only GAP1 is meaningful. It measures the time between the command or data word and the first (and only) status word in the message. For RT-to-RT messages, GAP2 measures the time between the last data word and the second status word. The Gap Times Word bit definitions are as shown in Figure [10-21](#).

<b>msb</b>										<b>lsb</b>
15				8	7					0
GAP2					GAP1					

Figure 10-21. Gap Times Word bit definitions.

	<p>Gap measurements shall be made IAW MIL-STD-1553 response time measurements from the mid-bit zero crossing of the parity bit of the last word to the mid-zero crossing of the sync of the status word.</p>
---	--

10.6.4.2.1.2.3 Length Word. The Length of the message is the total number of bytes in the message. A message consists of command words, data words, and status words.

10.6.4.3 Packet Format. Unless an error occurred, as indicated by one of the error flags in the block status word, the first word following the length should always be a command word. The resultant packets have the format shown in Figure [10-22](#).

<b>msb</b>	<b>lsb</b>
15	0
PACKET HEADER	
CHANNEL SPECIFIC DATA (BITS 15-0)	
CHANNEL SPECIFIC DATA (BITS 31-16)	
INTRA-PACKET TIME STAMP FOR MSG 1 (BITS 15-0)	
INTRA-PACKET TIME STAMP FOR MSG 1 (BITS 31-16)	
INTRA-PACKET TIME STAMP FOR MSG 1 (BITS 47-32)	
INTRA-PACKET TIME STAMP FOR MSG 1 (BITS 63-48)	
INTRA-PACKET DATA HEADER FOR MSG 1 (BITS 15-0)	
INTRA-PACKET DATA HEADER FOR MSG 1 (BITS 31-16)	
INTRA-PACKET DATA HEADER FOR MSG 1 (BITS 47-32)	
COMMAND WORD	
COMMAND, STATUS, OR DATA WORD	
DATA OR STATUS WORD	
:	
DATA OR STATUS WORD	
INTRA-PACKET TIME STAMP FOR MSG 2 (BITS 15-0)	
INTRA-PACKET TIME STAMP FOR MSG 2 (BITS 31-16)	
INTRA-PACKET TIME STAMP FOR MSG 2 (BITS 47-32)	
INTRA-PACKET TIME STAMP FOR MSG 2 (BITS 63-48)	
INTRA-PACKET DATA HEADER FOR MSG 2 (BITS 15-0)	
INTRA-PACKET DATA HEADER FOR MSG 2 (BITS 31-16)	
INTRA-PACKET DATA HEADER FOR MSG 2 (BITS 47-32)	
COMMAND WORD	
COMMAND, STATUS, OR DATA WORD	
DATA OR STATUS WORD	
:	
DATA OR STATUS WORD	
:	
INTRA-PACKET TIME STAMP FOR MSG n (BITS 15-0)	
INTRA-PACKET TIME STAMP FOR MSG n (BITS 31-16)	
INTRA-PACKET TIME STAMP FOR MSG n (BITS 47-32)	
INTRA-PACKET TIME STAMP FOR MSG n (BITS 63-48)	
INTRA-PACKET DATA HEADER FOR MSG n (BITS 15-0)	
INTRA-PACKET DATA HEADER FOR MSG n (BITS 31-16)	
INTRA-PACKET DATA HEADER FOR MSG n (BITS 47-32)	
COMMAND WORD	
COMMAND OR DATA, WORD	
DATA OR STATUS WORD	
:	
DATA OR STATUS WORD	
PACKET TRAILER	

Figure 10-22. MIL-STD-1553 Data Packet, Format 1.

10.6.5 Analog Data Packets, Format 1. An illustration of the generic packet structure for analog data is shown in Figure 10-23. An Analog Data Packet will contain a Channel Specific Data word for each subchannel of analog data sampled within that packet. This will be followed by at least one complete sampling schedule of data. A sampling schedule is defined as a sampling sequence in which each subchannel, described by a Channel Specific Data word, is sampled at least once. In many cases, due to simultaneous sampling rules and varied sampling rates (see 10.6.5.2), a particular subchannel will be sampled more than once during a sampling schedule. In addition, multiple complete sampling schedules may be included in a single packet. For these reasons, the number of Channel Specific Data words will usually be less than the number of samples. Figure 10-23 depicts the generic packet data structure for M data subchannels and a single sampling schedule that has a length N. Note that the width of the structure is not related to any number of bits and is merely presented to show the relative placement of words within the packet.

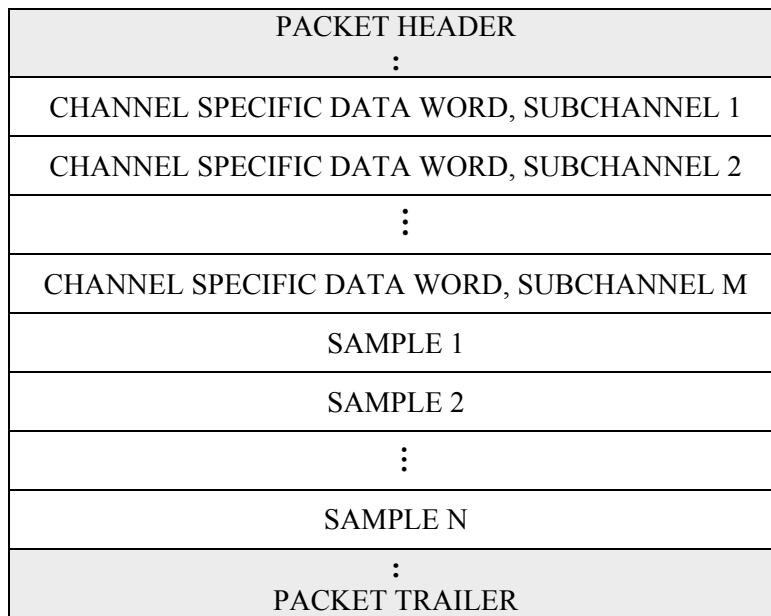



Figure 10-23. Generic Analog Data Packet, Format 1.

 <p><b>NOTE</b></p>	<p>The Packet Header Time in an Analog Data Packet shall correspond to the first data sample in the packet. There are no Intra-Packet Headers in Analog Data Packets.</p>
--	---

10.6.5.1 Analog Packet Channel Specific Data. The Packet Body portion of each Analog Packet begins with the Channel Specific Data Word(s). Each subchannel that is sampled within the packet sampling schedule must have a Channel Specific Data word within the packet. Channel Specific Data words for Analog Data Packets are formatted as shown in Figure [10-24](#).

<b>msb</b>						<b>lsb</b>					
31	28	27	24	23	16	15	8	7	2	1	0
RESERVED		FACTOR		TOTCHAN		SUBCHAN		LENGTH		MODE	

Figure 10-24. Analog Data Packets, Channel Specific Data word format.

where:

Bits 1-0: Mode: indicates alignment and packing modes of the analog data. Bit 0 is the packing bit, bit 1 is the alignment bit (X = don't care). When Totchan (defined below) is more than 1, the Mode must be the same for all subchannels in a single packet.

- X0 = data is packed
- 01 = data is unpacked, lsb padded
- 11 = data is unpacked, msb padded

Bits 7-2: Length: indicates a binary value representing the number of bits in the analog to digital converter (A/D).

- 000000 = sixty-four bit samples
- 000001 = one bit samples
- :
- :
- 001000 = eight bit samples
- :
- 001100 = twelve bit samples
- :

Bits 15-8: Subchan: indicates a binary value representing the number of the analog subchannel. When an analog packet contains data from more than one subchannel, the Channel Specific Data words must be inserted into the packet in ascending subchannel number as identified by this Subchan field. The Subchan values in these Channel Specific Data words need not be contiguous (see Totchan) but they must be in ascending decimal numerical order with the exception that subchannel 0 (256) is last.

- 0x01 = Subchannel 1
- 0x02 = Subchannel 2
- :
- 0x00 = Subchannel 256

Bits 23-16: Totchan: indicates the total number of analog subchannels in the packet (and the number of Channel Specific Data words in the packet). This Totchan field must be the same value in all Channel Specific Data words in a single packet. The Totchan value may be less than the largest subchan value. This can happen when a multi-channel analog input device has some of its subchannels disabled (turned off) for a specific recording. For example, if an analog input device has eight subchannels and not all eight are active, an analog data packet may have three subchannels (Totchan = 3) numbered 4, 7, and 8 (enabled Subchan = 4, 7, 8.) The number of subchannels (Totchan) and the subchannel number for each active subchannel (Subchan) in a packet are identified in the accompanying TMATS (Computer Generated Data, Format 1) packet.

0x00 = 256 subchannels

0x01 = 1 subchannel

0x02 = 2 subchannels

:

Bits 27-24: Factor: is the exponent of the power of 2 sampling rate factor denominator for the corresponding subchannel (described in [10.6.5.2](#)) in the range 0 to 15. (The sampling rate factor numerator is always 1.)

0x0 = sampling rate factor denominator 20 = 1 => factor = 1/1

0x1 = sampling rate factor denominator 21 = 2 => factor = 1/2

0x2 = sampling rate factor denominator 22 = 4 => factor = 1/4

:

0xF = sampling rate factor denominator 215 = 32768 => factor = 1/32768

Bits 31-28: Reserved.

10.6.5.2 Analog Samples. To preserve timing relationships and allow for accurate reconstruction of the data, a simultaneous sampling scheme shall be employed. The highest sampling rate required shall define the primary simultaneous sampling rate within the packet. The primary simultaneous sampling rate is identified in the Telemetry Attributes Transfer Standard (TMATS) file describing the attributes of the analog data packet. The rate at which the other subchannels are sampled is then defined by the sampling factor (1, 1/2, 1/4, 1/8, 1/16, .....1/32768) for each subchannel. As an example, a sampling factor of 1/4 would yield that subchannel being sampled at one-fourth the primary simultaneous sampling rate and a sampling factor of 1 would yield that subchannel being sampled at the primary simultaneous sampling rate.

Directly following the Channel Specific Data word(s), at least one complete sampling schedule shall be inserted in the packet. The samples, within the sampling sequence, may be inserted either unpacked, MSB-Packed, or LSB-Packed as described in paragraph [10.6.5.2.1](#) and [10.6.5.2.2](#). In either case, one or more subchannels may be included in a single packet. When



multiple subchannels are encapsulated into a single packet, the subchannel with the highest sampling rate requirement defines the primary simultaneous sampling rate. The rate at which the other subchannels are sampled is defined by the sampling factor (contained within the Channel Specific Data words). Sampling factors are defined as:

$$\left(\frac{1}{2^K}\right) * X$$

where:

$K = 0, 1, 2, 3, 4, 5, \dots$  of the primary simultaneous sampling rate,  $X$ .

The subchannels are then sampled and ordered such that the highest sample rate:

$1 * X$  subchannel(s) appear in every simultaneous sample,

$\left(\frac{1}{2}\right) * X$  in every 2nd simultaneous sample,

$\left(\frac{1}{4}\right) * X$  in every 4th simultaneous sample

and so on until all the subchannels are sampled, resulting in a complete sampling schedule of all subchannels described by the Channel Specific Data words. In doing so, the total number of simultaneous samples (not the total number of samples) will equal the denominator of the smallest sampling factor and all subchannels are sampled in the last simultaneous sample.

For example, a packet with six subchannels with Sampling Factors  $1/2, 1/8, 1, 1/2, 1,$  and  $1/8$ , respectively, will yield a sampling sequence within the data packet as follows:

Simultaneous Sample 1: Subchannel 3  
 Simultaneous Sample 1: Subchannel 5

Simultaneous Sample 2: Subchannel 1  
 Simultaneous Sample 2: Subchannel 3  
 Simultaneous Sample 2: Subchannel 4  
 Simultaneous Sample 2: Subchannel 5

Simultaneous Sample 3: Subchannel 3  
 Simultaneous Sample 3: Subchannel 5

Simultaneous Sample 4: Subchannel 1  
 Simultaneous Sample 4: Subchannel 3  
 Simultaneous Sample 4: Subchannel 4  
 Simultaneous Sample 4: Subchannel 5

Simultaneous Sample 5: Subchannel 3

Simultaneous Sample 5:	Subchannel 5
Simultaneous Sample 6:	Subchannel 1
Simultaneous Sample 6:	Subchannel 3
Simultaneous Sample 6:	Subchannel 4
Simultaneous Sample 6:	Subchannel 5
Simultaneous Sample 7:	Subchannel 3
Simultaneous Sample 7:	Subchannel 5
Simultaneous Sample 8:	Subchannel 1
Simultaneous Sample 8:	Subchannel 2
Simultaneous Sample 8:	Subchannel 3
Simultaneous Sample 8:	Subchannel 4
Simultaneous Sample 8:	Subchannel 5
Simultaneous Sample 8:	Subchannel 6

Notice that the denominator of the smallest sampling factor defined the number of simultaneous samples within the packet (in this example 8). However, the total number of samples within the sampling schedule does not have to equal the number of simultaneous samples (in this example 26). Also notice that all subchannels are sampled during the last simultaneous sample. The order of the subchannel samples in each Simultaneous Sample is ascending by subchannel number.

Any number of **complete** sampling schedules may be placed within a packet so that the maximum packet length is not exceeded. The TMATS file identifies the number of samples contained within each packet.

10.6.5.2.1 Unpacked Mode. In Unpacked Mode, packing is disabled, and each sample is padded with the number of bits necessary to align each word with the next 16-bit boundary in the packet. Four pad bits are added to 12-bit words, eight pad bits are added to 8-bit words, etc. All pad bits shall be zero.

To illustrate msb packing, given M analog subchannels mapping into N samples for the special case of all samples having bit lengths of 12 bits, the resultant analog packets with msb padding have the form shown in Figure [10-25](#).

<b>msb</b>	<b>lsb</b>
15	0
PACKET HEADER	
⋮	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 1 (BITS 15-0)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 1 (BITS 31-16)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 2 (BITS 15-0)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 2 (BITS 31-16)	
⋮	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL M (BITS 15-0)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL M (BITS 31-16)	
⋮	
4-PAD BITS	SAMPLE 1, 12-DATA BITS
4-PAD BITS	SAMPLE 2, 12-DATA BITS
4-PAD BITS	SAMPLE 3, 12-DATA BITS
⋮	
4-PAD BITS	SAMPLE N, 12-DATA BITS
⋮	
PACKET TRAILER	

Figure 10-25. Analog Data Packet – msb Unpacked Mode.

To illustrate lsb packing, given M analog subchannels mapping into N samples for the special case of all samples having bit lengths of 12 bits, the resultant analog packets with lsb padding have the form shown in Figure [10-26](#).

<b>msb</b>	<b>lsb</b>
15	0
PACKET HEADER	
⋮	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 1 (BITS 15-0)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 1 (BITS 31-16)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 2 (BITS 15-0)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 2 (BITS 31-16)	
⋮	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL M (BITS 15-0)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL m (BITS 31-16)	
⋮	
SAMPLE 1, 12-DATA BITS	4-PAD BITS
SAMPLE 2, 12-DATA BITS	4-PAD BITS
SAMPLE 3, 12-DATA BITS	4-PAD BITS
⋮	
SAMPLE N, 12-DATA BITS	4-PAD BITS
⋮	
PACKET TRAILER	

Figure 10-26. Analog Data Packet – lsb Unpacked Mode.

10.6.5.2.2 Packed Mode. In packed mode, packing is enabled and padding is not added to each data word. However, if the number of bits in the packet are not an integer multiple of 16, then ‘Y’ Filler bits will be used to msb fill the last data word to force alignment on a 16-bit boundary. ‘Y’ is sixteen (16) minus the integer remainder of L, the total number of data bits in the packet, divided by 16 and is mathematically expressed as

$$Y = 16 - (\text{MODULUS}\{L,16\})$$

To illustrate msb filling, given M analog subchannels mapping into N samples for the special case of all samples having bit lengths of 12 bits, the resultant analog packets with Filler bits at the end of the Nth sample have the form shown in Figure 10-27.

<b>msb</b>	<b>lsb</b>
15	0
PACKET HEADER	
:	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 1 (BITS 15-0)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 1 (BITS 31-16)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 2 (BITS 15-0)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL 2 (BITS 31-16)	
:	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL M (BITS 15-0)	
CHANNEL SPECIFIC DATA WORD, SUBCHANNEL M (BITS 31-16)	
SAMPLE 2 (BITS 3-0)	SAMPLE 1 (BITS 11-0)
SAMPLE 3 (BITS 7-0)	SAMPLE 2 (BITS 11-4)
:	:
Y FILLER BITS	SAMPLE N (BITS 11-0)
:	
:	
PACKET TRAILER	

Figure 10-27. Analog Data Packet – Packed Mode packet.

10.6.6 Discrete Data Packets, Format 1. A packet with Discrete data has the basic structure shown in Figure 10-28. Note that the width of the structure is not related to any number of bits. This drawing is merely to show the relative placement of data in the packet. One to 32 discrete states may be recorded for each event.

PACKET HEADER
CHANNEL SPECIFIC DATA
INTRA-PACKET HEADER FOR EVENT 1
EVENT 1 STATES
INTRA-PACKET HEADER FOR EVENT 2
EVENT 2 STATES
:
INTRA-PACKET HEADER FOR EVENT N
EVENT N STATES
PACKET TRAILER

Figure 10-28. General Discrete Data Packet, Format 1.

10.6.6.1 Discrete Packet Channel Specific Data Word. The Packet Body portion of each Discrete Packet begins with the Channel Specific Data word, which is shown in Figure [10-29](#).

<b>msb</b>						<b>lsb</b>
31		8	7		3	2
						0
	RESERVED		LENGTH		MODE	

Figure 10-29. Discrete packet Channel Specific Data word format.

where:

Bits 2-0: Mode indicates the mode of accessing the discrete data.

Bit 0 indicates the Record State.

- 0 = discrete data is recorded when the state changes
- 1 = discrete data is recorded on a time interval basis

Bit 1 indicates the alignment of the data.

- 0 = lsb
- 1 = msb

Bit 2 is reserved

Bits 7-3: Length: indicates a binary value representing the number of bits in the event.

Bits 31-8: Reserved.

10.6.6.2 Discrete Data. After the Channel Specific Data, the Discrete Data is inserted in the packet. Discrete data is described as Events (Figure 10-30). Each event includes the Event State for each discrete input and the corresponding intra-packet time. The event state is a 32-bit word that provides the logical state of each discrete input.

<b>msb</b>										<b>lsb</b>	
31	30	...						1	0		
D31	D30	...						D1	D0		

Figure 10-30. Discrete Data Event State word format.

where:

Bits 31-0: Discrete Event Bits indicate the states of the discrete event bits.

Bit 0 indicates Discrete 0 (D0) state.

0 = discrete 0 is at state 0.  
1 = discrete 0 is at state 1.

Bit 1 indicates Discrete 1 (D1) state.

0 = discrete 1 is at state 0.  
1 = discrete 1 is at state 1.

⋮

Bit 31 indicates Discrete 31 (D31) state.

0 = discrete 31 is at state 0.  
1 = discrete 31 is at state 1.

10.6.6.2.1 Discrete Event Intra-Packet Header. All discrete events shall include an Intra-Packet Header consisting of an Intra-Packet Time Stamp only, which is inserted immediately before the discrete event. The length of the Intra-Packet Header is fixed at 8 bytes (64-bits) positioned contiguously in the sequence shown in Figure 10-31.

<b>msb</b>										<b>lsb</b>	
31		15	12	11						0	
TIME (LSLW)											
TIME (MSLW)											

Figure 10-31. Discrete Event Intra-Packet Data Header.

10.6.6.2.1.1 Intra-Packet Time Stamp. This frame (8 bytes) indicates the time tag of the discrete event shown in Figure 10-32. First long word bits 31-0 and second long word bits 31-0 indicate the following values:

- The Relative Time Counter that corresponds to the first data bit of the discrete event with bits 31 to 16 in the second long word zero filled; or
- Time, if enabled by bit 7 in the Packet Header Flags (paragraph 10.6.1.1.7), corresponds to the time format indicated by bits 2 and 3 in the Packet Secondary Header Time format (paragraph 10.6.1.1.7) and to the first data bit of the discrete event.

msb 15	lsb 0
PACKET HEADER	
CHANNEL SPECIFIC DATA (BITS 15-0)	
CHANNEL SPECIFIC DATA (BITS 31-16)	
INTRA-PACKET TIME STAMP FOR EVENT 1 (BITS 15-0)	
INTRA-PACKET TIME STAMP FOR EVENT 1 (BITS 31-16)	
INTRA-PACKET TIME STAMP FOR EVENT 1 (BITS 47-32)	
INTRA-PACKET TIME STAMP FOR EVENT 1 (BITS 63-48)	
STATES FOR EVENT 1 (BITS 15-0)	
STATES FOR EVENT 1 (BITS 31-16)	
:	
INTRA-PACKET TIME STAMP FOR EVENT n (BITS 15-0)	
INTRA-PACKET TIME STAMP FOR EVENT n (BITS 31-16)	
INTRA-PACKET TIME STAMP FOR EVENT n (BITS 47-32)	
INTRA-PACKET TIME STAMP FOR EVENT n (BITS 63-48)	
STATES FOR EVENT n (BITS 15-0)	
STATES FOR EVENT n (BITS 31-16)	
PACKET TRAILER	

Figure 10-32. Discrete Data – Packet format.

10.6.7 Computer Generated Data Packets. Packets with Computer Generated data (Meta, ASCII) have the basic structure shown in Figure 10-33. Formats 0 and 1 are used to add information packets to recorded data. This information contains annotation data and setup information for the data that is recorded. Note that the width of the structure is not related to any number of bits. This figure merely represents the relative placement of data in the packet.

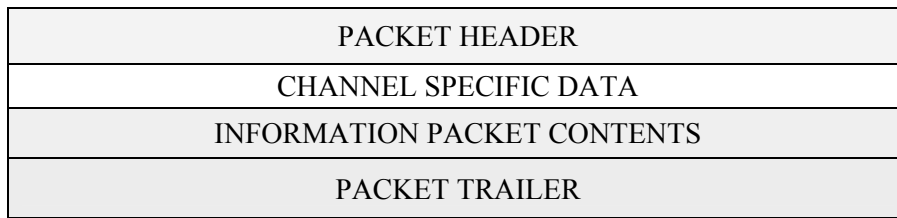


Figure 10-33. General Computer Generated Data Packet format.

10.6.7.1 Computer Generated Packet Channel Specific Data Word. The Packet Body portion of each Computer Generated Data Packet begins with the channel Specific Data word, which is formatted as shown in Figure [10-34](#).



Figure 10-34. Computer Generated Data Packet - Channel Specific Data word format.

10.6.7.2 Computer Generated Packet Data. After the Channel Specific Data, the Computer Generated Data is inserted in the packet. The organization and content of the Computer Generated Data is determined by the specific format type. There are no Intra-Packet Headers with Computer Generated Data Packets.

10.6.7.2.1 Format 0 – User Defined Data. Format 0 enables the insertion of user-defined, Computer Generated Data (Meta, ASCII).


10.6.7.2.2 Format 1 – Setup Records. Format 1 defines a setup record that describes the hardware, software, and data channel configuration used to produce the other data packets in the file. The organization and content of a Format 1 Setup Record is IAW with IRIG 106 Chapter 9 TMATS standard. It is mandatory for a TMATS record to be utilized to configure the solid-state recorder. A Format 1 Computer Generated Data Packet containing the TMATS record utilized to configure the recorder shall be the first packet in each data file.

10.6.8 ARINC-429 Data Packets, Format 0. Data shall be packetized in Word Mode: each 32-bit word of an ARINC-429 bus shall be preceded by an Intra-Packet Header containing an Intra-Packet Data Header only with an identifier (ID WORD) that provides type and status information. The Intra-Packet Header does not contain an Intra-Packet Time Stamp. The packet time in the packet header is the time of the first ARINC data word in the packet, and the time of successive ARINC data words is determined from the first word time using the gap times in the ID words that precede each of the data words. Multiple words of multiple ARINC-429 buses can be inserted into a single packet. The resultant packets shall have the format indicated in Figure [10-35](#).



<b>msb</b> 15	<b>lsb</b> 0
PACKET HEADER	
CHANNEL SPECIFIC DATA (BITS 15-0)	
CHANNEL SPECIFIC DATA (BITS 31-16)	
ID WORD FOR DATA WORD 1	
ID WORD FOR DATA WORD 1	
ARINC-429 DATA WORD 1 (BITS 15-0)	
ARINC-429 DATA WORD 1 (BITS 31-16)	
ID WORD FOR DATA WORD 2	
ID WORD FOR DATA WORD 2	
ARINC-429 DATA WORD 2 (BITS 15-0)	
ARINC-429 DATA WORD 2 (BITS 31-16)	
:	
ID WORD FOR DATA WORD n	
ID WORD FOR DATA WORD n	
ARINC-429 DATA WORD n (BITS 15-0)	
ARINC-429 DATA WORD n (BITS 31-16)	
PACKET TRAILER	

Figure 10-35. ARINC-429 Data Packet format.

 <p><b>NOTE</b></p>	<p>Time tagging of ARINC-429 shall correspond to the first data bit of the packet.</p>
--	--

10.6.8.1 ARINC-429 Packet Channel Specific Data Word. The Packet Body portion of each ARINC-429 data packet shall begin with a Channel Specific Data word formatted as shown in Figure [10-36](#).

<b>msb</b> 31	16 15	<b>lsb</b> 0
RESERVED		MSGCOUNT

Figure 10-36. ARINC 429 Packet Channel Specific Data Word format.

where:

Bits 15-0: Message count (MSGCOUNT): indicates the binary value of the number of ARINC-429 words included in the packet.

Bits 31-16: Reserved.

10.6.8.2 Intra-Packet Data Header. Bits 31-0 contain the ARINC-429 ID WORD. Each ARINC-429 bus data word is preceded by an identification word and the bit definitions are as shown in Figure [10-37](#).

<b>msb</b>								<b>lsb</b>
31	24	23	22	21	20	19		0
SUBCHANNEL-1		FE	PE	BS	RS	GAP TIME		

Figure 10-37. ARINC 429 ID Word bit definitions.

where:

- Bits 19-0: Gap Time: contains a binary value that represents the gap time from the beginning of the preceding bus word to the beginning of this bus word in 0.1 microsecond increments. The gap time of the first word in the packet is GAP TIME = 0. Before total time for packet data reaches 100 milliseconds, a new packet must be started.
- Bit 20: Reserved.
- Bit 21: ARINC-429 Bus (BS) indicates which ARINC-429 bus the data is from.  
 0 = Indicates Low-Speed ARINC-429 bus (12.5 kHz).  
 1 = Indicates High-Speed ARINC-429 bus (100 kHz).
- Bit 22: Parity Error (PE) indicates an ARINC-429 parity error.  
 0 = No parity error has occurred.  
 1 = Parity error has occurred.
- Bit 23: Format Error (FE) indicates an ARINC-429 format error.  
 0 = No Format Error Has Occurred.  
 1 = Format Error Has Occurred.
- Bits 31-24: Subchannel indicates a binary value that defines the ARINC-429 channel number belonging to the following data word. '0' means first channel.

10.6.8.3 ARINC-429 Packet Data Words. ARINC-429 data words shall be inserted into the packet in the original 32-bit format as acquired from the bus.

10.6.9 Message Data Packets, Format 0. The data from one or more separate serial communication interface channels can be placed into a message data packet as shown in Figure [10-38](#).

<b>msb</b> 15	<b>lsb</b> 0
PACKET HEADER	
CHANNEL SPECIFIC DATA (BITS 15-0)	
CHANNEL SPECIFIC DATA (BITS 31-16)	
INTRA-PACKET TIME STAMP FOR MSG 1 (BITS 15-0)	
INTRA-PACKET TIME STAMP FOR MSG 1 (BITS 31-16)	
INTRA-PACKET TIME STAMP FOR MSG 1 (BITS 47-32)	
INTRA-PACKET TIME STAMP FOR MSG 1 (BITS 63-48)	
INTRA-PACKET DATA HEADER FOR MSG 1 (BITS 15-0)	
INTRA-PACKET DATA HEADER FOR MSG 1 (BITS 31-16)	
BYTE 2	BYTE 1
:	:
FILLER (IF n IS ODD)	BYTE n
:	
INTRA-PACKET TIME STAMP FOR MSG n (BITS 15-0)	
INTRA-PACKET TIME STAMP FOR MSG n (BITS 31--16)	
INTRA-PACKET TIME STAMP FOR MSG n (BITS 47-32)	
INTRA-PACKET TIME STAMP FOR MSG n (BITS 63-48)	
INTRA-PACKET DATA HEADER FOR MSG n (BITS 15-0)	
INTRA-PACKET DATA HEADER FOR MSG n (BITS 31-16)	
BYTE 2	BYTE 1
:	:
FILLER (IF n IS ODD)	BYTE n
PACKET TRAILER	

Figure 10-38. Message Data Packet format.

10.6.9.1 Message Packet Channel Specific Data Word. The packet body portion of each Message Data Packet begins with a Channel Specific Data word. It defines if the Packet Body contains several short messages (type: *complete*) or one segment of a long message (type: *segmented*).

10.6.9.1.1 Complete Message Channel Specific Data Word. The Channel Specific Data word is formatted for the Complete type of packet body as shown in Figure [10-39](#).

<b>msb</b> 31	18	17	16	15	<b>lsb</b> 0
RESERVED	TYPE		COUNTER		

Figure 10-39. Complete Message Channel Specific Data Word Format.

where:

Bits 15-0: Counter: contains a binary value that represents the number of messages included in the packet.

Bits 17-16: Type: indicates the type of Serial Packet.

00 = One or more complete messages.

01 = Reserved.

10 = Reserved.

11 = Reserved.

Bits 31-18: Reserved.

10.6.9.1.2 Segmented Message Channel Specific Data Word. The Channel Specific Data word is formatted for the Segmented type of packet body as shown in Figure [10-40](#).

<b>msb</b>						<b>lsb</b>
31		18	17	16	15	0
RESERVED			TYPE		COUNTER	

Figure 10-40. Segmented type of Packet Body Channel Specific Data Word format.

where:

Bits 15-0: Counter: contains a binary value that represents the segment number of a long message. The number must start with 1, and must be incremented by one after each packet. The maximum length of a single long message can be 4 Gbytes (combined with the 16-bit Message Length field, see 10.6.9.2.1 number of messages included in the packet).

Bits 17-16: Type: indicates the type of Serial Packet.

00 = Reserved.

01 = Packet is a beginning of a long message from a single source.

10 = Whole packet is the last part of a long message from a single source.

11 = Whole packet is a middle part of a long message from a single source.

Bits 31-18: Reserved.

10.6.9.2. Message Data Intra-Packet Header. After the Channel Specific Data, Message Data is inserted into the packet. Each Message is preceded by an Intra-Packet Header that has both an Intra-Packet Time Stamp and an Intra-Packet Data Header containing a Message ID Word. The length of the Intra-Packet Header is fixed at 12 bytes (96-bits) positioned contiguously in the sequence shown in Figure [10-41](#).

<b>msb</b>	<b>lsb</b>
31	0
15	
TIME (LSLW)	
TIME (MSLW)	
MESSAGE ID WORD	

Figure 10-41. Message Data Intra-Packet Header.

10.6.9.2.1 Intra-Packet Time Stamp. This frame (8 bytes) indicates the time tag of the Message Data. First long word bits 31-0 and second long word bits 31-0 indicate the following values:

- The Relative Time Counter that corresponds to the first data bit in the message with bits 31 to 16 in the second long word zero filled; or
- Packet Secondary Header Time Type, if enabled by bit 6 in the Packet Header Flags (paragraph 10.6.1.1.7), corresponds to the time format indicated by bits 2 and 3 in the Packet Secondary Header Time format (paragraph [10.6.1.1.7](#)) and to the first data bit in the Message.

10.6.9.2.2 Intra-Packet Data Header. The Intra-Packet Data Header is an identification word (Message ID Word) that precedes the message and is inserted into the packet with the format shown in Figure [10-42](#).

<b>msb</b>		<b>lsb</b>	
31	30	29	0
		16	15
DE	FE	SUBCHANNEL - 1	MESSAGE LENGTH

Figure 10-42. Intra-Packet Data Header format.

where:

- Bits 15-0: Message Length: contains a binary value that represents the length of the message in bytes (n) that follows the ID Word. The maximum length of a message (complete) or a message segment (segmented) is 64K bytes.
- Bits 29-16: Subchannel: contains a binary value that represents the subchannel number belonging to the message that follows the ID Word when the Channel ID in the packet header defines a group of subchannels. Zero means first and/or only sub-channel.

Bit 30: Format Error (FE): used to indicate a protocol error, such as out-of-sequence data or length errors.

0 = No Format Error.  
1 = Format Error encountered.

Bit 31: Data Error (DE): used to indicate bad data bits as determined by parity, checksums, or CRC words received with the data.

0 = No Data Error has occurred.  
1 = Data Error has occurred.

10.6.10 Video Packets, Format 0 (MPEG-2). Format 0 Video Packets uses the industry standard MPEG-2 Main Profile @ Main Level (MP@ML) and Transport Stream Frames (TSF) per ISO/IEC 13818-1:2000 (see Table 10-6). These two MPEG algorithm features are combined to produce an encoded video stream, which can be encapsulated using conventional IRIG-106 Chapter 4 PCM techniques. This encapsulation method will be specified in this section as it pertains to Format 0 Mpeg-2 Video Packets.

By utilizing MP@ML, which is currently the most common combination of MPEG-2 profiles and levels, it is possible to code an ITU-R 601 recorder picture format without filtering processes before coding. This will eliminate the need for proprietary encoding/decoding (CODEC) filters which would violate the intent of an “open” standard and make decoding of the data difficult without specific knowledge of or access to the encoding process.

TABLE 10-6. MP@ML ALGORITHMS			
Profile Table		Level Table	
B-frames	YES	Maximum Bit Rate	15 Mbps
Chroma_format	4:2:0	Buffer Size	1835008 bits
Scalability	NONE	Maximum Sample Density	720 samples/lines 576 lines/frame 30 frames/s
Intra DC precision	8, 9, 10 bits	Luminance Sample Rate	10368000
		Horizontal Vector Range	-512:+511.5
		Vertical Vector Range (frame pictures)	-128:+127.5

A packet with Format 0 MPEG-2 Video data has the basic structure shown in Figure 10-43. Note that the width of the structure is not related to any number of bits. This figure merely represents the relative placement of data in the packet.

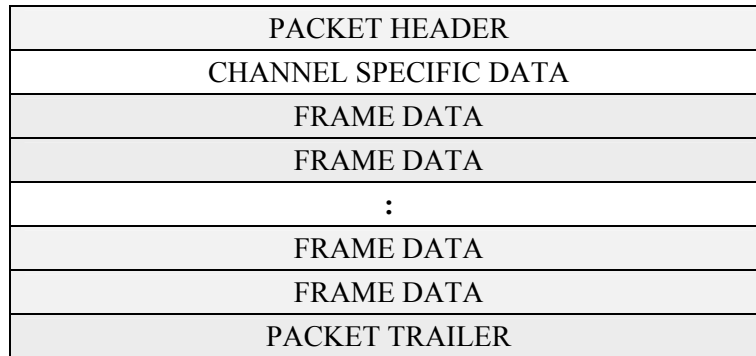


Figure 10-43. General MPEG-2 Video Packet, Format 0.

10.6.10.1 Video Packet Audio. When recording video using Format 0 MPEG-2 Video, and audio is also required data, audio will be inserted into the TSF per ISO/IEC 13818-3. A separate analog channel to specifically record audio is not required as MPEG-2 supports audio insertion into the TSF. By combining video and audio recording, bandwidth and memory capacity will be increased.

10.6.10.2 Video Packet Channel Specific Data Word. The packet body portion of each format 0 MPEG-2 video packet begins with the Channel Specific Data word, which is formatted as shown in Figure 10-44.

<b>msb</b>							<b>lsb</b>	
31	30	29	28	27	24	23	18	17
R	R	MA	MI	LOCKST	MODE	RESERVED		
								0

Figure 10-44. Video Packet Channel Specific Data Word format.

where:

Bits 17-0: Reserved.

Bits 23-18: Mode: indicates the Data Packing Mode (ref [10.6.2.2.1](#) and [10.6.2.2.2](#))

Bits 23-20 are reserved.

Bit 19 indicates Packed Data Mode.

0 = Packed Data Mode not enabled.

1 = Packed Data Mode enabled.

Bit 18 is reserved.

Bits 27-24: Lock Status (LOCKST): indicates the lock status of the frame synchronizer.

Bits 27-26 indicate Minor Frame Status.

00 = Reserved.

01 = Reserved.

10 = Minor Frame Check (after losing Lock).

11 = Minor Frame Lock.

Bits 25-24 indicate Major Frame Status.

00 = Minor Frames only.

01 = Reserved.

10 = Major Frame Check (after losing Lock).

11 = Major Frame Lock.

Bit 28: Minor Frame Indicator (MI): indicates if the first word in the packet is the beginning of a TSF.

0 = First word is not the beginning of a minor frame.

1 = First word is the beginning of a minor frame.

Bit 29: Major Frame Indicator (MA): indicates if the first word in the packet is the beginning of a major TSF.

0 = First word is not the beginning of a major frame.

1 = First word is the beginning of a major frame.

Bits 31-30: Reserved (R).

10.6.10.3 Video Packet Data. A Format 0 Video Packet shall contain an integral number of Transport Stream Frames (TSFs.) No Intra-Packet Headers are inserted in Format 0 Video Data Packets (Figure [10-46](#)). The Packet Header time is the time of the first TSF in the packet. The bit rate of the encoding will be user selectable and within the MP@ML specification, but the frame format must be set up as follows for proper alignment of the TSF:

- a. 94 words per frame (includes sync)
- b. 16 bits per word
- c. 8 bit sync pattern, 01000111 (0x47)

A TSF is made up of fixed-length 188 byte frames containing an 8-bit sync pattern or “sync byte” (starting at bit 0 and ending at bit 7 of the TSF). The sync bytes value is 01000111 (0x47). The first bit (bit 0) of the first word in the PCM frame will be aligned on the first bit (bit 0) of the TSF sync byte. The rest of the TSF 187 data bytes will follow as shown in Fig. [10-45](#).



<b>msb</b>	<b>lsb</b>
15	0
TSF DATA BITS 15 TO 8	TSF SYNC BYTE BITS 7 TO 0
TSF DATA BITS 31 TO 16	
:	
TSF DATA BITS 1503 TO 1488	

Figure 10-45. Format 0 MPEG-2 Video Frame Sync and Word format.

<b>msb</b>	<b>lsb</b>
15	0
PACKET HEADER	
CHANNEL SPECIFIC DATA (BITS 15-0)	
CHANNEL SPECIFIC DATA (BITS 31-16)	
TSF DATA BITS 15 TO 0	
TSF DATA BITS 31 TO 16	
:	
TSF DATA BITS 1487 TO 1472	
TSF DATA BITS 1503 TO 1488	
TSF DATA BITS 15 TO 0	
TSF DATA BITS 31 TO 16	
:	
TSF DATA BITS 1487 TO 1472	
TSF DATA BITS 1503 TO 1488	
:	
REPEAT FOR EACH TSF	
:	
PACKET TRAILER	

Figure 10-46. Format 0 MPEG-2 Video Data – sample packet.

10.6.11 Image Packets, Format 0. A Format 0 Image Packet shall contain one or more fixed-length segments of one or more video images (Figure 10-47). The Channel Specific Data word for an image packet identifies the number of segments in the packet and the portion of the image or images contained in the packet. If the optional Intra-Packet Header is not included with each segment, the Relative Time Counter in the packet header is the time of the first segment in the packet.

<b>msb</b>	<b>lsb</b>
15	0
PACKET HEADER	
CHANNEL SPECIFIC DATA (BITS 15-0)	
CHANNEL SPECIFIC DATA (BITS 31-16)	
OPTIONAL INTRA-PACKET HEADER FOR SEGMENT 1 (BITS 15-0)	
OPTIONAL INTRA-PACKET HEADER FOR SEGMENT 1 (BITS 31-16)	
OPTIONAL INTRA-PACKET HEADER FOR SEGMENT 1 (BITS 47-32)	
OPTIONAL INTRA-PACKET HEADER FOR SEGMENT 1 (BITS 63-48)	
BYTE 2	BYTE 1
:	:
FILLER (IF n IS ODD)	BYTE n
:	
OPTIONAL INTRA-PACKET HEADER FOR SEGMENT n (BITS 15-0)	
OPTIONAL INTRA-PACKET HEADER FOR SEGMENT n (BITS 31-16)	
OPTIONAL INTRA-PACKET HEADER FOR SEGMENT n (BITS 47-32)	
OPTIONAL INTRA-PACKET HEADER FOR SEGMENT n (BITS 63-48)	
BYTE 2	BYTE 1
:	:
FILLER (IF n IS ODD)	BYTE n
PACKET TRAILER	

Figure 10-47. Image Packet, Format 0.

10.6.11.1 Image Packet Channel Specific Data Word. The Packet Body portion of each Image Packet begins with a Channel Specific Data word. It defines the byte length of each segment and indicates if the Packet Body contains several complete images or partial images, and whether or not the Intra-Packet Data Header precedes each segment (Figure 10-48).

<b>msb</b>					<b>lsb</b>
31	30	29	28	27	26
PARTS	SUM	IPH	LENGTH		
					0

Figure 10-48. Image Packet Channel Specific Data Word format.

where:

- Bits 26-0: Length: indicates a binary value that represents the byte length of each segment.
- Bit 27: Intra-Packet Header (IPH): indicates that the Intra-Packet Header (Time Stamp) precedes each segment of the image.  
 0 = Intra-Packet Header not enabled.  
 1 = Intra-Packet Header enabled.
- Bits 29-28: Sum: indicates if the packet contains a partial image, one complete image, multiple complete images, or pieces from multiple images.  
 00 = Packet contains less than one complete image.  
 01 = Packet contains one complete image.  
 10 = Packet contains multiple complete images.  
 11 = Packet contains multiple incomplete images.
- Bits 31-30: Parts: indicates which piece or pieces of the video frame are contained in the packet.  
 00 = Packet does not contains first or last segment of image.  
 01 = Packet contains first segment of image.  
 10 = Packet contains last segment of image.  
 11 = Packet contains both first and last segment of image.

10.6.11.2 Image Intra-Packet Header. After the Channel Specific Data, Format 1 Image Data is inserted into the packet. Each block of data is optionally preceded by an Intra-Packet Header as indicated by the IPH bit in the Channel Specific Data word. When included, the Intra-Packet Header consists of an Intra-Packet Time Stamp only. The length of the Intra-Packet Header is fixed at 8 bytes (64-bits) positioned contiguously in the sequence shown in Figure [10-49](#).

<b>msb</b>	<b>lsb</b>
31	0
15	
TIME (LSLW)	
TIME (MSLW)	

Figure 10-49. Format 1 Image Data Intra-Packet Data Header.

10.6.11.2.1 Intra-Packet Time Stamp. This frame (8 bytes) indicates the time tag of the Format 0 Image Data. First long word bits 31-0 and Second long word bits 31-0 indicate the following values:

- The Relative Time Counter that corresponds to the first data bit in the first byte with bits 31 to 16 in the second long word zero filled; or

- Packet Secondary Header Time Type, if enabled by bit 6 in the Packet Header Flags (paragraph 10.6.1.1.7), corresponds to the time format indicated by bits 2 and 3 in the Packet Secondary Header Time (paragraph [10.6.1.1.7](#)) and to the first data bit in the Message.

10.6.12 UART Data Packets, Format 0. The data from one or more separate asynchronous serial communication interface channels (RS-232, RS-422, RS-485, etc.) can be placed into a UART Data Packet as shown in Figure [10-50](#).

<b>msb</b>	<b>lsb</b>
15	0
PACKET HEADER	
CHANNEL SPECIFIC DATA (BITS 15-0)	
CHANNEL SPECIFIC DATA (BITS 31-16)	
(OPTIONAL) INTRA-PACKET DATA HEADER FOR UART 1 (BITS 15-0)	
(OPTIONAL) INTRA-PACKET DATA HEADER FOR UART 1 (BITS 31-16)	
(OPTIONAL) INTRA-PACKET DATA HEADER FOR UART 1 (BITS 47-32)	
(OPTIONAL) INTRA-PACKET DATA HEADER FOR UART 1 (BITS 63-48)	
UART ID for UART 1 (BITS 15-0)	
UART ID for UART 1 (BITS 31-16)	
BYTE 2	BYTE 1
:	:
FILLER (IF n IS ODD)	BYTE n
:	
(OPTIONAL) INTRA-PACKET DATA HEADER FOR UART n (BITS 15-0)	
(OPTIONAL) INTRA-PACKET DATA HEADER FOR UART n (BITS 31-16)	
(OPTIONAL) INTRA-PACKET DATA HEADER FOR UART n (BITS 47-32)	
(OPTIONAL) INTRA-PACKET DATA HEADER FOR UART n (BITS 63-48)	
UART ID for UART n (BITS 15-0)	
UART ID for UART n (BITS 31-16)	
BYTE 2	BYTE 1
:	:
FILLER (IF n IS ODD)	BYTE n
PACKET TRAILER	

Figure 10-50. UART Data Packet Format 0.

10.6.12.1 UART Packet Channel Specific Data Word, Format 0. The Packet Body portion of each UART Data Packet begins with a Channel Specific Data word as shown in Figure [10-51](#).

msb	lsb
31	0
IPH	RESERVED

Figure 10-51. UART Packet Channel Specific Data Word, Format 0.

where:

Bits 30-0: Reserved.

Bit 31: Intra-Packet Header (IPH): indicates that the Intra-Packet Header is inserted before the UART ID Words.

0 = Intra-Packet Header not enabled.

1 = Intra-Packet Header enabled.

10.6.12.2 UART Intra-Packet Header, Format 0. After the Channel Specific Data, UART Data is inserted into the packet. Each block of data is preceded by an Intra-Packet Header consisting of the Intra-Packet Time Stamp and a UART ID Word Intra-Packet Data Header. The length of the Intra-Packet Header is fixed at 8 bytes (64-bits) positioned contiguously in the sequence shown in Figure [10-52](#).

<b>msb</b>		<b>lsb</b>
31	15	0
TIME (LSLW)		
TIME (MSLW)		
UART ID WORD		

Figure 10-52. UART Data Intra-Packet Data Header.

10.6.12.2.1 Intra-Packet Time Stamp. This frame (8 bytes) indicates the time tag of the Format 1 Image Data. First long word bits 31-0 and second long word bits 31-0 indicate the following values:

- The Relative Time Counter that corresponds to the first data bit in the first byte with bits 31 to 16 in the second long word zero filled; or
- Packet Secondary Header Time Type, if enabled by bit 6 in the Packet Header Flags (paragraph 10.6.1.1.7), corresponds to the time format indicated by bits 2 and 3 in the Packet Secondary Header Time (paragraph [10.6.1.1.7](#)) and to the first data bit in the Message.

10.6.12.2.2 Intra-Packet Data Header. The Intra-Packet Data Header is an identification word (UART ID Word) that precedes the data and is inserted into the packet with the format indicated in Figure [10-52](#).

<b>msb</b>									<b>lsb</b>
31	30		29		16	15			0
PE	RESERVED	SUBCHANNEL - 1				DATA LENGTH			

Figure 10-53. UART Data ID Word format.

where:

- Bits 15-0: Data Length: indicates a binary value that represents the length of the UART data in bytes (n) that follow the UART ID Word.
- Bits 29-16: Subchannel: indicates a binary value that defines the subchannel number belonging to the data that follows the UART ID Word when the Channel ID in the packet header defines a group of subchannels. Zero means first and/or only sub-channel that the Intra-Packet Data Header is inserted before the UART ID Words.
- Bit 30: Reserved.
- Bit 31: Parity Error (PE): indicates a Parity Error.  
0 = No Parity Error  
1 = Parity Error

## 10.7 Solid State Recorder Control and Status

10.7.1 Recorder Control. The recorder may be controlled by either discrete control/status lines and/or serial communication ports. The serial interface shall consist of both RS-232 and RS-422 full duplex serial communications.

10.7.2 Communication Ports. The RS-232 and RS-422 serial communication ports shall be functional simultaneously without requiring selection of either port. Status requested by either port shall be returned on both ports. Note that unexpected results may occur if commands are issued on both ports simultaneously.

10.7.3 RS-232 Port. An RS-232 port shall be available at the Download Port.

10.7.4 Commands. Commands received through the serial communication ports shall not override hardware discrete controls.

10.7.5 Status Requests. Status requests received through the serial communication ports shall not interfere with hardware controls.

10.7.6 Serial Status. Serial status shall be provided on either serial status request or discrete activation.

10.7.7 Default Interface. Default Interface with user equipment shall utilize the following ASCII serial communication protocol:

- 38400 baud
- One start bit
- 8 bit data
- No parity
- One stop bit

10.7.8 Serial Commands. The following SSR commands are a subset of the Recorder Command and Control Mnemonics defined in IRIG Standard 106 Chapter 6 Section 18, where additional rules regarding command syntax and recorder operation are also specified, along with examples showing the use of each command. Appendix A is a copy of Chapter 6, Section 18. The SSR commands are simple ASCII command strings delimited by spaces. All commands begin with an ASCII period (“.”) and, with the single exception of the .TMATS command, end with a carriage return and line-feed terminator sequence. Table [6-15](#) (Appendix A) summarizes the required commands.

10.7.9 Required Discrete Control Functions. Required discrete control functions are noted in Figure [10-54](#).

Description
RECORD
ERASE
DECLASSIFY
ENABLE
BIT

Figure 10-54. Required Discrete Control functions.

10.7.9.1 Control and Status Lines. In addition to the five contacts for discrete control, five lines for indicating status shall be provided. Grounding a control line (or causing the indicator line to go to ground) referenced to the recorder’s ground completes the circuit to activate a function (Figure [10-55](#)).

10.7.9.1.1 RECORD Command. Activated by toggle switch (normally closed position .55 volts or less), this discrete commands the recorder to start recording. Recorder will remain in this mode until such time as the switch is set to normally open position.

10.7.9.1.2 ERASE Command. Activated by momentary switch (.55 volts or less, minimum duration of 100 ms), this discrete commands the recorder to erase its user data and file directory memory provided the enable switch is also activated.

10.7.9.1.3 DECLASSIFY Command. Activated by momentary switch (.55 volts or less, minimum duration of 100 ms), this discrete causes the recorder to start the declassify procedure provided the enable switch is also activated.

10.7.9.1.4 Command ENABLE. Activated by momentary switch (.55 volts or less) for either ERASE or DECLASSIFY discrete to operate.

10.7.9.1.5 BIT Command. Activated by momentary switch (.55 volts or less), this discrete commands the recorder to start the BIT procedure.

10.7.9.1.6 Record Status. A “record” indication (ON) shall be active at .55 volts or less. A “non-record” indication (OFF) will be an open circuit. Current limit of 60 milliamps required.

10.7.9.1.7 BIT Status. A “BIT” indication (ON) shall be .55 volts or less. A “non-BIT” indication (OFF) will be an open circuit. Current limit of 60 milliamps required.

10.7.9.1.8 Fault Status. A “fault” indication (ON) shall be .55 volts or less. A “non-fault” indication (OFF) will be an open circuit. Current limit of 60 milliamps required.

10.7.9.1.9 Erase Status. An “erase” indication (ON) shall be .55 volts or less. A “non-erase” indication (OFF) will be an open circuit. Current limit of 60 milliamps required.

10.7.9.1.10 Declassify Status. A “declassify” indication (ON) shall be .55 volts or less. A “non-declassify” indication (OFF) will be an open circuit. No discrete control line shall be available at the download port. Current limit of 60 milliamps required.

10.7.10 Voltage. Auxiliary voltage output of 28 Vdc shall be provided from the discrete/control port (250 mA maximum, short circuit protection).

10.7.11 Status Querying. Status querying shall be limited to intervals not to exceed two seconds and not faster than one second.



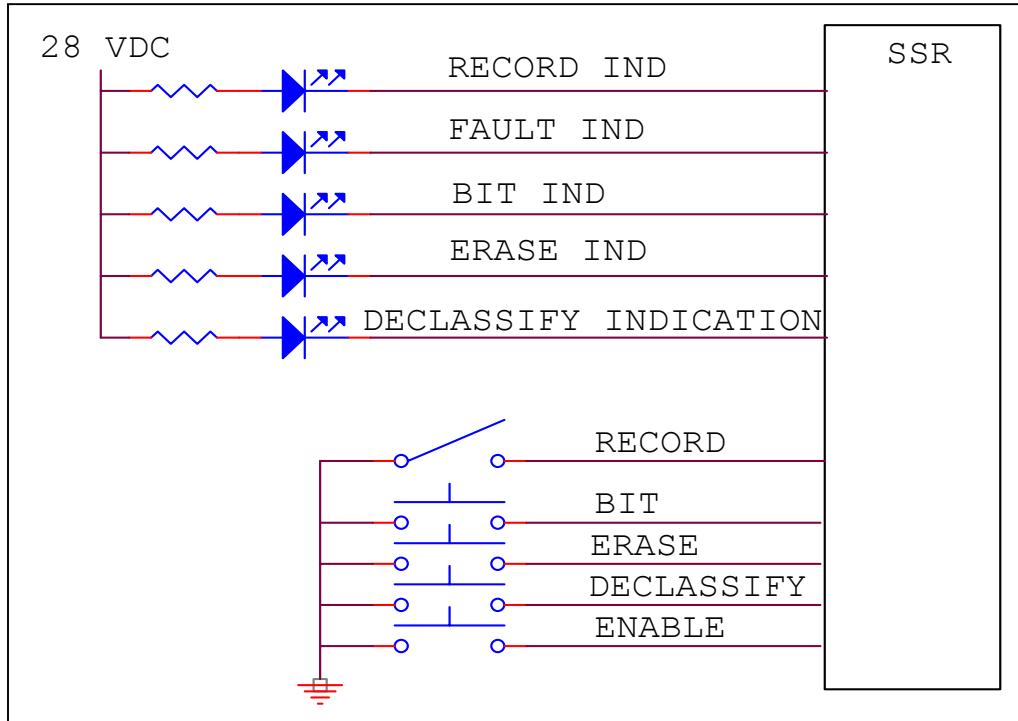


Figure 10-55. Discrete Control and Indicator functional diagram.

## 10.8 Declassification

10.8.1 Associated Documents. Documents such as NSA-130-2, DOD 5200.28 (1972) and DCI-116 historically covered declassification guidelines/requirements. These documents focused on declassification of standard disk and other conventional memory technologies. With the advent of advanced, high-density memory technologies, additional guidance must be provided. A new document that addresses various solid state, hard disk, floppy disk, RAID, and other storage media, declassification is being developed under NTISSP-9 working group for U.S. Policy.

10.8.2 Approach. The following approaches for declassification are currently recommended. The risk that proper declassification has been effectively implemented will reside ultimately with the user/customer/program manager. It is believed that the user is the most qualified to determine the declassification procedures for any program situation. It is their responsibility to correctly apply the guidelines to the program in each location to optimize the cost/effect while providing appropriate protection for the data. The guidelines are planned to be available on the Internet at Defense Link.

10.8.3 Algorithm. The algorithm to erase secure data is described in the sections below. During the Secure Erase procedure, all blocks of memory shall be processed. No block in memory shall be excluded from Secure Erase processing for any reason.

10.8.3.1 First Erase. Every memory block on the board is erased. Any erase failures reported by memory chips will result in the corresponding chip/block being declared a bad block. In the

event this bad block is not already in the corresponding board's bad block table, a new bad block entry will be appended onto the board's bad block table. Note that this new entry will not have the Secure Erase flag set.

10.8.3.2 First Write (0x55). Every memory chip location is recorded with the pattern 0x55. As each location is written, the data is read back to guarantee that all bits were written to the expected pattern. Any write failures reported by the chips, or any data errors will result in the corresponding chip/block being declared a bad block. In the event this bad block is not already in the corresponding board's bad block table, a new bad block entry will be appended onto the board's bad block table. Note that this new entry will not have the Secure Erase flag set.

10.8.3.3 Second Erase. Every memory chip shall be erased. Any erase failures reported by the memory chips will result in the corresponding chip/block being declared a bad block. In the event this bad block is not already in the corresponding board's bad block table, a new bad block entry will be appended onto the board's bad block table. Note that this new entry will not have the Secure Erase flag set.

10.8.3.4 Second Write (0xAA). Every memory chip location is recorded with the pattern 0xAA. As each location is written, the data is read back to guarantee that all bits were written to the expected pattern. Any write failures reported by the memory chips, or any data errors will result in the corresponding chip/block being declared a bad block. In the event this bad block is not already in the corresponding board's bad block table, a new bad block entry will be appended onto the board's bad block table. Note that this new entry will not have the Secure Erase flag set.

10.8.3.5 Third Erase. Every memory location is erased. Any erase failures reported by the memory chips will result in the corresponding chip/block being declared a bad block. In the event this bad block is not already in the corresponding board's bad block table, a new bad block entry will be appended onto the board's bad block table. Note that this new entry will not have the Secure Erase flag set.

10.8.3.6 Usable Secure Erased Blocks. All blocks that do not have an entry in the bad block table are now considered to be Secure Erased.

10.8.3.7 Unusable Secure Erased Blocks. If a bad block entry contains the flag indicating it has already been secure erased, this block has already been Secure Erased and requires no further processing, since it is known that this block was skipped during the previous recording.

10.8.3.8 Unsecure Bad Block Processing. A board's bad block table may contain bad block entries that have not previously been Secure Erased. If any such entries exist, the following steps are performed on each block.

10.8.3.8.1 Write Zeros Loop. For each page in the block, a pattern of all zeros is written to the page, and the page is checked to determine if any unexpected ones (UOs) are found. If any UOs are found, the page is re-written to all zeros. This process is repeated up to 16 times. After all

allowed re-writes, the board, chip, and block numbers of the block containing any remaining UOs are written to a “Failed Erase” Table.

10.8.3.8.2 Write Ones Loop. For each page in the block, the page is erased (to all ones) and checked to determine if any unexpected zeros (UZs) are found. If any UZs are found, another erase command is issued to the block. This process is repeated up to 16 times. After all allowed erase operations, the board, chip, and block numbers of the block containing any remaining UZs are written to the Failed Erase Table.

10.8.3.9 Failed Erase Table Processing. Any remaining entries in the Failed Erase Table correspond to blocks that cannot be erased. These blocks may still contain user data and, therefore, are declared to have failed the Secure Erase.

A count of the number of bad blocks in the Failed Erase Table that have not been Secure Erased is returned as part of the Secure Erase results. A non-zero count indicates a Secure Erase failure of at least one block. A command will allow the user to retrieve the Failed Erase Table. A command will also allow a user to retrieve the data from such blocks and manually determine if these blocks can be designated as “Secure Erased.” In most cases a single stuck bit will not compromise any user data and the offending block can be manually declared to be Secure Erased. If the results of manual inspection are indeterminate, the chip containing the failed block must be removed and destroyed, and the Secure Erase procedure must be repeated.

10.8.3.10 Secure Erase Completion. When all blocks are secure erased (no entries in the Failed Erase Table), a single file is written that completely fills the memory. The content of the file is the ASCII string “Secure Erase” repeated over and over. The name of the file in the file table is “SecureErase.”

## APPENDIX A

### IRIG STANDARD 106-03, *TELEMETRY STANDARD*

#### CHAPTER 6

#### MAGNETIC TAPE RECORDER AND REPRODUCER STANDARD

#### SECTION 6.18

#### RECORDER COMMAND AND CONTROL MNEMONICS (CCM)

##### 6.18 Recorder Command and Control Mnemonics (CCM)

This Section describes a set of standard commands and replies that can control tape, disk, and solid-state recorders. Not all commands may be applicable to all types of recorders or recorder implementations. Manufacturers who claim compliance with this Standard shall identify in an Interface Control Document for each recorder model the specific command and reply subset that is implemented. An important aspect of the CCM standard is the required command-response protocol. For each command issued to a recorder, there shall be exactly one response from the recorder, and the response shall begin immediately upon conclusion of the command input. There shall be no delay between the receipt of the command at the recorder and the transmission of the reply by the recorder. Commands that initiate recorder functions, which require time to complete, shall be replied to immediately, and the host shall poll the recorder status to determine when the function is complete. There shall be no unsolicited status output from the recorder, with one exception. This exception is a boot message upon leaving the POWER ON state, notifying the host that the recorder is ready to accept commands. The boot command shall contain a single asterisk as the last character. Thereafter, the recorder will only output in response to a command input. (A hardware reset or a software .RESET shall return the recorder to the POWER ON state.)

6.18.1 Recorder State Transitions. Figure 6-18 is a generic state transition diagram for standard recorder operation. Upon application of power, the recorder enters the POWER ON state, during which commands are not accepted. Upon conclusion of the power-up sequence, the recorder shall execute a built-in test (BIT) to verify recorder functionality. Upon successful conclusion of the BIT, the recorder shall enter the IDLE state. The following facts describe and explain the state transition diagram.

- a. The STARTING and STOPPING (ENDING) states may require zero (none) or more wait states, as necessary, for a particular recorder and command implementation.
- b. Some recorders can record without playing, play without recording, or record and play at the same time.
- c. For those recorders that require data clocks, the record clock is always external (provided by the source of the data). The playback clock, on the other hand, may be externally or

internally supplied, and when externally supplied, may or may not be synchronous to (equal to or derived from) the record clock.

- d. Some functions are implemented using multiple commands. For example, a conventional longitudinal recorder shuttle command is implemented as a .FIND command with the starting point identifier, followed by a .SHUTTLE command with the ending point identifier. Once the initial .SHUTTLE command is received, the recorder automatically initiates a FIND sequence when the end point is reached, and then automatically initiates a PLAY sequence when the start point is found. This is shown on the state transition diagram as the decision box “ANOTHER COMMAND PENDING.”
- e. Some recorders are physically able to record over existing data. This standard prevents recording over existing data by forcing the record point to the current end of data (EOD). An erase command is provided to enable reuse of the media by resetting the record point to the beginning of media (BOM.)
- f. Some recorders are physically able to replay data in either the forward sequence or reverse sequence. Forward is the sequence in which the data was recorded, whereas reverse is the opposite sequence. This standard only requires and supports replay in the forward sequence.

6.18.2 Command Summary. All commands must comply with the following syntax rules:

- a. All recorder commands are simple ASCII character strings delimited by spaces.
- b. All commands begin with an ASCII period (“.”) and, with the single exception of the .TMATS command, end with the first occurrence of a carriage return and line-feed terminator sequence.
- c. Parameters are separated from the commands and from each other with ASCII space characters.
- d. With one exception, command words and parameters may not include spaces. The one exception is the [*text string*] parameter for the .EVENT command.
- e. Multiple consecutive terminators and extraneous space characters are ignored.
- f. Each command is followed with either a simple response and an ASCII asterisk (“\*”) response terminator or the asterisk response terminator only, indicating the recorder is ready for the next command.
- g. All numeric parameters, with one exception, are decimal numbers. The one exception is the [*mask*] parameter for the .CRITICAL command, which is hexadecimal.
- h. Three commands, .FIND, .REPLAY, and .SHUTTLE, have numeric parameters that required units of measure. The [*mode*] parameter is used to specify the unit of measure (time, feet, or blocks.) If the [*mode*] parameter is omitted, the recorder shall use the most recently entered [*mode*].
- i. A [*time*] parameter value has five parts: days, hours, minutes, seconds, and milliseconds. Any part not entered defaults to zero except days, which defaults to don’t care (current day.) A period (“.”) identifies the start of the millisecond part, a hyphen (“-” separates the day from the hours, and colon characters (“:”) separate the hours, minutes, and

seconds. The following are valid times: 123- (day only), 17 (hours only), 17:30 (hours and minutes), 17:30:05 (hours, minutes, seconds), 17:0:05 (hours, minutes, seconds), 17:30:05.232 (hours, minutes, seconds, milliseconds), 123-17 (day, hours), 123-17:30 (day, hours, minutes), etc.

The available commands are summarized in Table [6-15](#).

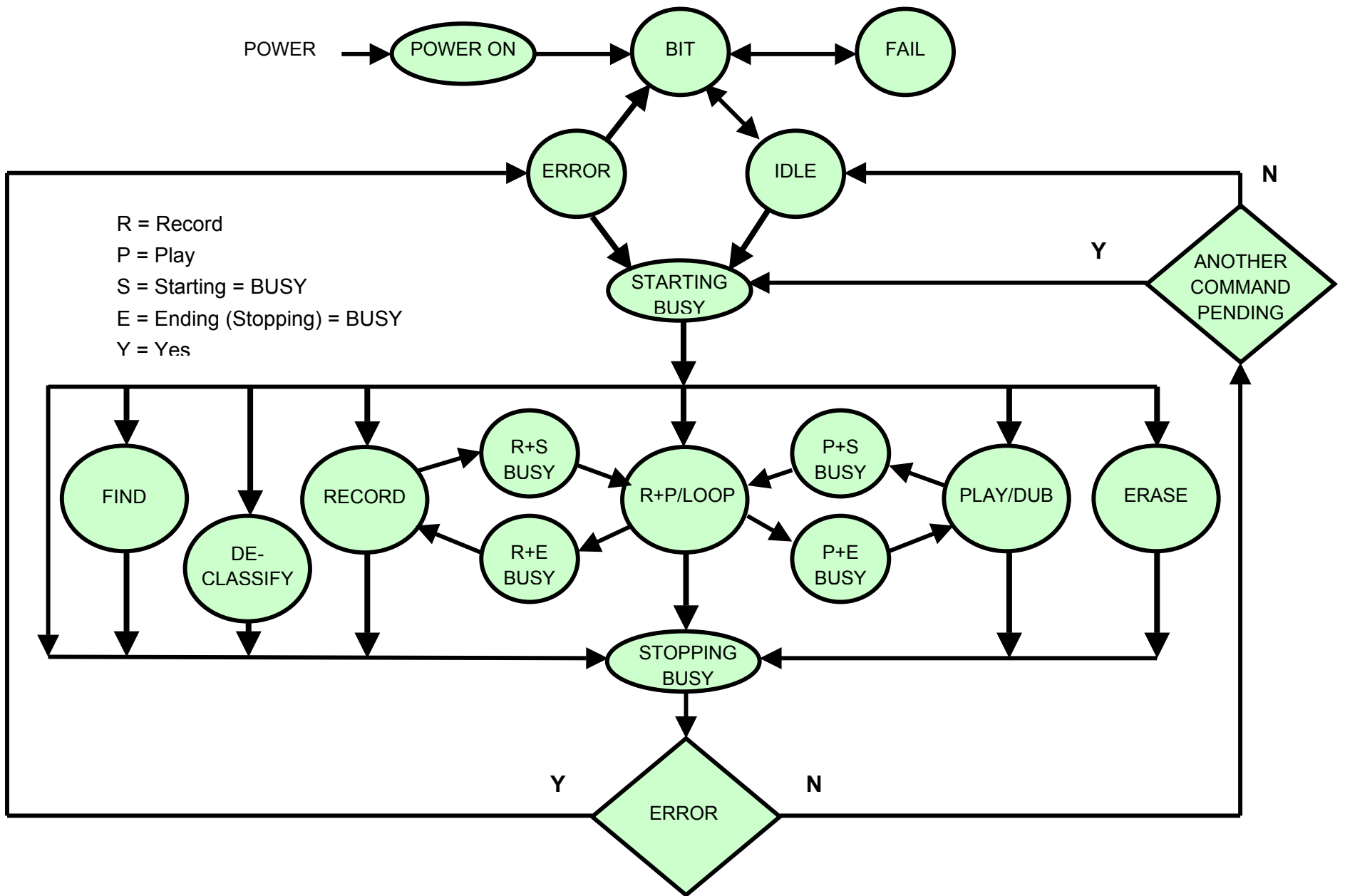


Figure 6-18. Recorder state transition diagram.

TABLE 6-15 COMMAND SUMMARY		
Command	Parameters <sup>1</sup>	Description
.BIT		Runs all of the built-in-tests
.CRITICAL	[ <i>n</i> [ <i>mask</i> ] ]	Specify and view masks that determine which of the .HEALTH status bits are critical warnings
.DECLASSIFY		Secure erases the recording media
.DISMOUNT		Unloads the recording media
.DUB	[ <i>location</i> ]	Same as .PLAY but with internal clock
.ERASE		Erases the recording media
.EVENT	[ <i>text string</i> ]	Display event table or add event to event table
.FILES		Displays information about each recorded file.
.FIND	[ <i>value</i> [ <i>mode</i> ] ]	Display current locations or find new play point
.HEALTH	[ <i>feature</i> ]	Display detailed status of the recorder system
.HELP		Displays table of "dot" commands
.LOOP		Starts record and play in read-after-write mode
.MEDIA		Displays media usage summary
.MOUNT		Powers and enables the recording media
.PLAY	[ <i>location</i> ]	Reproduce recorded data starting at [ <i>location</i> ] using external clock
.RECORD	[ <i>filename</i> ]	Starts a recording at the current end of data
.REPLAY	[ <i>endpoint</i> [ <i>mode</i> ] ]	Same as .SHUTTLE but with internal clock
.RESET		Perform software initiated system reset
.SETUP	[ <i>n</i> ]	Displays or selects 1 of 16 (0...15) pre-programmed data recording formats
.SHUTTLE	[ <i>endpoint</i> [ <i>mode</i> ] ]	Play data repeatedly from current location to the specified endpoint location using external clock
.STATUS		Displays the current system status
.STOP	[ <i>mode</i> ]	Stops the current recording, playback, or both
.TIME	[ <i>start-time</i> ]	Displays or sets the internal system time
.TMATS	{ <i>mode</i> } [ <i>n</i> ]	Write, Read, Save, or Get TMATS file

**Note:**

1. Parameters in braces “{}” are required. Parameters in brackets “[ ]” are optional. When optional parameters are nested (“[xxx [yy] ]”), the outer parameter (xxx) must be specified in order to also specify the inner parameter (yy).



6.18.3 Command Error Codes. Issuing invalid commands (bad syntax) or illegal commands (not accepted in the current system state) result in error code responses prior to the asterisk response terminator when a command cannot be completed. Table 6-16 shows possible error codes and the conditions under which they occur.

TABLE 6-16. COMMAND ERROR CODES		
Error	Description	Conditions
00	INVALID COMMAND	Command does not exist
01	INVALID PARAMETER	Parameter is out of range, or wrong alpha-numeric type
02	INVALID MODE	Command cannot be executed in the current state
03	NO MEDIA	Recording media is dismounted or not installed
04	MEDIA FULL	Command cannot be executed because there is no free space available on the recording media
05	COMMAND FAILED	Command failed to execute for any reason other than those listed above

The error message is displayed before the asterisk response terminator with an ASCII “E” identifier.

Example:

```
. RECORD
E 03
*
```

6.18.4 Command Parameters and Responses. Each of the commands, the command parameters, and the recorder responses to the commands are described in the following sections.

6.18.4.1 .BIT: The **.BIT** command runs the built-in test (BIT) on the recorder. The prompt is returned immediately after the test is started. The **.BIT** command is only valid in the IDLE, ERROR, and FAIL states. During the BIT, the user must periodically check the status until the test is complete. While in BIT mode, the percent completion is shown with the **.STATUS** command. The result of the **.BIT** command is go/no-go status indicated by the end state. If the system returns to the IDLE state, the BIT was successful. If the system goes to the FAIL state, the BIT failed and further system-specific diagnostics are required. The ASCII “S” in the response is the identifier of a **.STATUS** response.

Example:

```
.BIT
* .STATUS
S 02 0 0 21%
* .STATUS
S 02 0 0 74%
* .STATUS
S 01 0 0
*
```



Parameters in braces “{}” are required. Parameters in brackets “[ ]” are optional. When optional parameters are nested (“[xxx [yy] ]”), the outer parameter (xxx) must be specified in order to also specify the inner parameter (yy).

6.18.4.2 **.CRITICAL** [*n* [*mask* ]]: The **.CRITICAL** command is used to view and specify the critical warning masks used with the **.HEALTH** command. An encoded 32-bit status word is displayed with the **.HEALTH** command for each feature in the recorder. The **.CRITICAL** command allows the user to specify which status word bits constitute critical warnings. If a bit in the **.CRITICAL** mask word for a feature is set, then the corresponding **.HEALTH** status word bit for that feature signals a critical warning. The **.CRITICAL** command without any parameters returns the mask word for each feature in ascending feature order. The **.CRITICAL** command with a single parameter, the feature number, returns the list of descriptive warning strings and status word bit associations for the specified feature. The **.CRITICAL** command with both the feature number parameter and the 8-character ASCII hexadecimal mask value parameter specifies a new mask value for the feature. All mask values in the command responses are hexadecimal.

Example #1: The command with no parameters returns the mask for each feature.

```
.CRITICAL
1  FFFFFFFF
2  00000300
3  00000001
4  00000003
   :
   :
15 00000002
16 00000000
*
```


Example #2: The command with the feature number parameter only, no mask value, returns all of the possible warning text strings for the specified feature and shows which .HEALTH status word bit is associated with the particular warning.

```
.CRITICAL 4
00000001 No Clock
00000002 No Minor Frame Lock
00000004 Slow Clock
00000100 No Major Frame Lock
00000200 Sync Bit Error
*
```

Example #3: Entering both the feature number parameter and the mask value parameter resets the mask for the specified feature.

```
.CRITICAL 4 00000103
4 00000103
*
```

6.18.4.3 .DECLASSIFY: The **.DECLASSIFY** command erases all recorded data using an approved declassification procedure and sets the record point to the beginning of media (BOM).

	This command will permanently erase all recorded data. Data cannot be recovered once this command has been executed!
---	---

The prompt is returned immediately after the operation is started. During declassify, the user must periodically check the status until the operation is complete. While in DECLASSIFY state, the percent completion is shown with the .STATUS command.

Example:

```
.DECLASSIFY
* .STATUS
S 04 0 0 23%
* .STATUS
S 04 0 0 84%
* .STATUS
S 01 0 0
*
```

6.18.4.4 .DISMOUNT: The **.DISMOUNT** command disables and, if necessary, removes power from the active recording media. The media may be removed only after this command is issued.

Example #1:


```
.DISMOUNT
*
```

Example #2: If a failure occurs an error message is displayed before the prompt.

```
.DISMOUNT
E 03
*
```

6.18.4.5 .DUB [location]. The **.DUB** command is identical to the **.PLAY** command except that it specifies the use of the internal playback clock to retrieve the recorded data.

6.18.4.6 .ERASE: The **.ERASE** command erases all data and resets the record point to the beginning of media (BOM).

	<p>This command will permanently erase all recorded data. Data cannot be recovered once this command has been executed!</p>
---	---

The prompt is returned immediately after the operation is started. During erase, the user must periodically check the status until the operation is complete. While in ERASE state, the percent completion is shown with the .STATUS command.

Example:

```
.ERASE
*.STATUS
S 03 0 0 23%
*.STATUS
S 03 0 0 84%
*.STATUS
S 01 0 0
*
```

6.18.4.7 .EVENT [*text string*]. The **.EVENT** command adds an entry to the recorder event file or displays all of the current event file entries. If a non-blank text string is included with the command, a new event entry is appended to the event file with the text string in the message field of the event entry. The text string may be any length, but only the first 48 bytes, starting with the first non-blank character but including all subsequent blanks, are saved in the event file entry. If no text string is provided with the message, the current event file entries are displayed as a list of character strings showing the event sequence number, the absolute event time based on the recorder system time, the current media address (block number) at the time the event entry is created, and the optional text string.

Example:

```
.EVENT -
*.EVENT This text was supplied with the .EVENT command
*.EVENT x
*.EVENT
1 001-00:13:58.109 101231 -
2 001-00:14:11.106 433213 This text was supplied with the
.EVENT command
3 001-17:44:06.677 2427654 x
*
```

6.18.4.8 .FILES: The **.FILES** command displays a list of character strings showing information about each recording session (file). Each string in the list contains the file number, file name, starting block number, file size in bytes, start day, and start time of the file. For those systems that also store the end day and time of each file, the end day and time may be added to the end of each file string. File names may not contain space or asterisk characters. If user names are not assigned to individual recordings, the default file names shall be “file1,” “file2,” etc. Each file string shall be formatted as shown in the following example (with optional end day and end time).

Example:

```
.FILES
1  TPD-10  10000  272760832  001-00:13:58.109  001-00:14:03.826
2  TPD-11  92884  425984000  001-00:14:11.106  001-00:14:28.602
3  file3   350790  305430528  123-17:44:06.677  123-17:44:13.415
*
```

6.18.4.9 **.FIND** [*value* [*mode*]]: The **.FIND** command is used to report the current record and play point or to set the play point to the desired location within the recorded data. The desired location can be expressed in a number of different formats or “modes:” time, blocks, and feet. When the command is entered without any parameters, the recorder returns the current record point and current play points, using the current default mode. The default mode is declared each time a mode parameter is supplied with the **.FIND** command, the **.REPLAY** command, or the **.SHUTTLE** command. Thereafter, the mode parameter may be omitted and the recorder will use the default mode. The mode keywords are TIME, BLOCKS, and FEET.

The location specified in the value parameter of the **.FIND** command can be numeric or one of six keywords: BOM (beginning of media), BOD (beginning of data), EOD (end of data), EOM (end of media), BOF (beginning of file), and EOF (end of file.) These keywords may be used with or without a mode parameter. Numeric location values, whether accompanied by the mode keyword or not, must be valid for the specified or default mode. Blocks and feet are entered as decimal integer numbers. Time is entered as specified in [paragraph 6.18.2 item i](#).

Example #1: Display the current record point and play point. The default mode is blocks.

```
.FIND
F 1022312 BOD
*
```

Example #2: Find a specific time in the recorded data.

```
.FIND 15:33:12 TIME
*.STATUS
S 08 0 0 41%
*.STATUS
S 08 0 0 84%
*.STATUS
S 01 0 0
*.FIND
F 102-16:18:27.000 102-15:33:12.000
*
```

6.18.4.10 **.HEALTH** [*feature*]: The **.HEALTH** command provides a standard mechanism for vendor-specific status information to be conveyed to the user. Entering the command without the optional parameter displays a list of system-specific “features” and an encoded status word for each feature. Entering a decimal feature number parameter with the command decodes the status word for a single feature and displays a list of messages pertaining to the feature, one for each set bit in the status word. The choice of features, their ordering, their descriptions, their encoded status words, and their decoded message lists are all vendor specific. This standard only requires that the syntax of the responses to the **.HEALTH** command conform to the following rules:

- a. If no features are implemented, the response to a **.HEALTH** command is the response terminator asterisk.
- b. Implemented features are numbered consecutively starting with 1 and displayed in ascending numerical order.
- c. The description of a feature may not contain an asterisk character.
- d. The feature list response (no feature number parameter supplied with the command) is a sequence of text strings, each containing the decimal feature number, the 8-character ASCII hexadecimal representation of the 32-bit status word for the feature, a text feature description, and a carriage return and line feed terminator. The value of the 32-bit status word for a “healthy” feature shall be all zeros. If a feature is disabled, the 8-character ASCII hexadecimal string shall be replaced with eight ASCII hyphen “-” characters.
- e. The individual feature response (feature number parameter supplied with the command) is a sequence of descriptive text strings, one for each set bit in the feature status word. Each string is terminated with a carriage return and line feed.

The **.CRITICAL** command is used to specify and view the mask word for each feature that determines if a set **.HEALTH** status word bit adds to the total non-critical or critical warning counts displayed with the **.STATUS** command.

Example #1:

```
.HEALTH
1 00000000 Time Code Input
2 00000000 Voice Input
3 ----- PCM Input #1
4 00000103 PCM Input #2
   :
15 00000000 1553 Input #2
16 00000000 Temp Monitor
*
```

Example #2:

```
.HEALTH 4  
No Clock  
No Minor Frame Lock  
No Major Frame Lock  
*
```

6.18.4.11 .HELP: The **.HELP** command displays a list showing a summary of the serial "dot" commands and parameters.

Example:

```
.HELP  
.BIT  
.CRITICAL [n [mask]]  
.DECLASSIFY  
.DISMOUNT  
.DUB [location]  
.ERASE  
.EVENT [message]  
.FILES  
.FIND [value [mode]]  
.HEALTH [feature]  
.HELP  
.LOOP  
.MEDIA  
.MOUNT  
.PLAY [location]  
.RECORD [filename]  
.REPLAY [endpoint [mode]]  
.RESET  
.SETUP [n]  
.SHUTTLE [endpoint [mode]]  
.STATUS  
.STOP [mode]  
.TIME [start-time]  
.TMATS {mode} [n]  
*
```

6.18.4.12 .LOOP: The **.LOOP** command is used to put the recorder into read-after-write mode, recording and simultaneously playing back the recorded data. If the recorder is already recording when the **.LOOP** command is issued, the command starts the playback at the current record point without affecting the recording.



Example:

```
.STATUS
S 01 0 0
* .LOOP
* .STATUS
S 07 0 0 35%
*
```

6.18.4.13 .MEDIA: The **.MEDIA** command displays the media usage summary. It shows the number of bytes per block, the number of blocks used and the number of blocks remaining, respectively.

Example:

```
.MEDIA
MEDIA 32768 1065349 6756127
*
```

6.18.4.14 .MOUNT: The **.MOUNT** command applies power and enables the recording. For systems with multiple memory canisters or media cartridges, the effect of the **.MOUNT** command on each canister or media cartridge is defined in advance with vendor-specific commands.

Example:

```
.MOUNT
*
```

6.18.4.15 .PLAY [*location*]: The **.PLAY** command starts a playback of the data at either the current play point or at the location specified in the optional parameter with the command using the user's external data clock. The current play point is defined to be the media location immediately following the most recently played data. If no **.PLAY** command has been issued since recorder power-on, the current play point is the beginning of data. The location parameter has two forms, [block\_number] and [filename [block\_offset]]. If the first character of the location parameter is numeric, the entire parameter must be numeric and it specifies the block number address at which to start the playback. When the first character of the location parameter is alphabetic, the parameter is the filename to playback and a second optional parameter that specifies the numeric 0-origin block offset into the named file may be included with the **.PLAY** command. To begin playing at a location other than a block number or file, use the **.FIND** command to position the play point to the desired location.

Example:

```
.PLAY file1 250
*
```

6.18.4.16 .RECORD [*filename*]: The **.RECORD** command starts a new recording. The optional file name parameter is an ASCII string with up to eleven characters, beginning with an alphabetic character, and with no spaces or asterisks. If the file name parameter is omitted, the filename will be of the form “file*n*”, where *n* is the file number. The recording will continue until the recording media is full or until the **.STOP** command is issued.

Example:

```
.RECORD
*
```

6.18.4.17 .REPLAY [*endpoint* [*mode*]]: The **.REPLAY** command is identical to the **.SHUTTLE** command, except that it specifies that the internal clock is to be used to retrieve the data.

6.18.4.18 .RESET: The **.RESET** command performs a software-initiated reset of the recorder, returning the recorder to the power-on state.

Example:

```
.RESET
*
```

6.18.4.19 .SETUP [*n*]: The **.SETUP** command chooses one of 16 pre-defined setups stored in the recorder. The optional parameter is a one or two digit decimal setup number from 0 to 15. The current setup may be displayed by omitting the setup number parameter.

Example #1:

```
.SETUP
SETUP 10
*
```

Including the setup number changes the setting.

Example #2:

```
.SETUP 5
SETUP 5
*
```

6.18.4.21 **.SHUTTLE** [*endpoint* [*mode*]]: The **.SHUTTLE** command initiates a repeated playback from the current play point to the end point specified in the command, using an external clock to retrieve the data. The syntax of the endpoint parameter is identical to that of the **.FIND** command.

Example:

```
.SHUTTLE 1430 FEET
*
```

6.18.4.21 **.STATUS**: The **.STATUS** command displays the current state of the recorder and two counts. The first is the total number of non-critical warning bits currently set and the second is the total number of critical warning bits currently set. If the recorder is in any state other than FAIL, IDLE, BUSY, or ERROR, the command also displays a progress percentage, the meaning of which is dependent on the specific state. Whenever the recorder is transitioning between states and the transition is not instantaneous, the **.STATUS** command will return the BUSY state. The ERROR state is entered when the currently executing command does not complete successfully. For example, when a **.FIND** command is unable to locate the specified position on the media, the recorder transitions to the ERROR state. Table 6-17 shows the various states by numerical code and describes the meaning of the progress percentage for each state. An ASCII “S” character identifies a **.STATUS** command response.

TABLE 6-17. RECORDER STATES		
State Code	State Name	Progress Description
00	FAIL	---
01	IDLE	---
02	BIT	Percent complete
03	ERASE	Percent complete
04	DECLASSIFY	Percent complete
05	RECORD	Percent media recorded
06	PLAY	Percent recording played
07	RECORD & PLAY	Percent media recorded
08	FIND	Percent complete
09	BUSY	---
10	ERROR	---

Example #1:

```
. STATUS
S 03 0 0 84%
*
```

For states that do not have a progress indication, that field is omitted in the response.

Example #2:

```
* . STATUS
S 01 0 0
*
```

6.18.4.22 **.STOP** [*mode*]: The **.STOP** command stops a recording, playback, or both. The optional mode parameter may be either the word RECORD or the word PLAY. If the optional mode parameter is not specified, both recording and playing, or either of the two modes if other is not active, will be stopped. Using the parameter enables either recording or playing to be stopped without affecting the other when both are active.

Example #1:

```
. STOP
*
```

The current state can be displayed with the status command.

Example #2:

```
* . STATUS
S 07 0 0 26%
* . STOP PLAY
* . STATUS
S 05 0 0 26%
*
```

The **.STOP** command returns an error if the recorder is not in the appropriate state.

Example #3:

```
* . STATUS
S 01 0 0
* . STOP
E 02
*
```

6.18.4.23 `.TIME` [start-time]: The `.TIME` command displays or sets the internal systems time. The optional start-time parameter is formatted as shown in the example below. Without a parameter, this command displays the current system time. The timestamps recorded with user data are derived from this clock.

Example #1:

```
.TIME  
TIME 001-23:59:59.123  
*
```

To set the time, enter a value expressed in days, hours, minutes, seconds and milliseconds. For example:

```
.TIME 123-13:01:35  
TIME 123-13:01:35.000  
*
```

Trailing values and punctuation may be omitted (zero is default).

Example #1:

```
.TIME 123-  
TIME 123-00:00:00.000  
*
```

Example #2:

```
.TIME 15:31  
TIME 000-15:31:00.000  
*
```

Example #3:

```
.TIME 15:31:20  
TIME 000-15:31:20.000  
*
```

6.18.4.24 **.TMATS** {mode} [n]: The **.TMATS** command provides a vendor-independent mechanism for loading a setup file into the recorder and retrieving a setup file from the recorder. The required mode parameter must be one of the following four words: **WRITE**, **READ**, **SAVE**, or **GET**. Writing or reading a **TMATS** file transfers the file between the external host and the recorder's internal volatile memory buffer. Saving or getting a **TMATS** file transfers the file between the recorder's internal volatile memory buffer and the recorder's internal non-volatile setup file storage area. To store a new setup file in the recorder, the **.TMATS WRITE** command is first used to transfer the file to the recorder, followed by a **.TMATS SAVE [n]** command to store the file in non-volatile memory. The numeric setup file number parameter is not valid with the **.TMATS WRITE** command. When saving the file to non-volatile memory, the optional setup file number parameter may be entered to designate a specific setup number (see the **.SETUP** command.) If the setup file number parameter is not specified with the **.TMATS SAVE** command, the file number defaults to setup 0. The **.TMATS GET [n]** command performs the inverse of the **.TMATS SAVE** command, retrieving the specified or default (0) file from non-volatile to volatile memory within the recorder. The **.TMATS READ** command transfers the file currently in the recorder's volatile setup file buffer to the host.

Termination of the **.TMATS WRITE** command string is unique. All other command strings terminate with the first occurrence of a carriage return and line feed sequence. The **.TMATS WRITE** command string does not terminate until the occurrence of a carriage return and line feed pair followed by the word **END** and another carriage return and line feed pair.

Example #1: The **.TMATS WRITE** command includes the **TMATS** file followed by the word **END**.

```
.TMATS WRITE  
G\DSI\N=18;  
G\DSI-1:TimeInChan1;  
G\DSI-2:VoiceInChan1;  
G\DSI-3:1553Chan01;  
:  
:  
P-8\IDC8-1:0;  
P-8\ISF2-1:ID;  
P-8\IDC5-1:M;  
END  
*
```

Example #2: The .TMATS READ command returns the file currently in the volatile buffer.

```
.TMATS READ
G\DSI\N=18;
G\DSI-1:TimeInChan1;
G\DSI-2:VoiceInChan1;
G\DSI-3:1553Chan01;
      :
      :
P-8\IDC8-1:0;
P-8\ISF2-1:ID;
P-8\IDC5-1:M;
*
```

Example #3: The .TMATS SAVE command stores the file in the volatile buffer to the designated non-volatile file memory in the recorder.

```
.TMATS SAVE 3
*
```

Example #4: The .TMATS GET command retrieves the designated file from non-volatile file memory in the recorder and puts it in a buffer that can be read by the user.

```
.TMATS GET 3
*
```

6.18.5 Command Validity Matrix. Table [6-18](#) identifies the recorder states where each of the serial commands is valid. The legend at the bottom of the table explains the matrix entry codes. Two codes, 3 and 4, identify states in which the associated command may or may not be valid due to system-specific implementation. Recorder users should assume that a command is not supported in a system-specific state (code 3 or 4) unless the specific recorder's Interface Control Document assures that support is provided.

6.18.6 Required Command Subset. Table [6-19](#) identifies the minimum subset of commands that must be implemented for each recorder type to be compliant with this standard.

**TABLE 6-18. COMMAND VALIDITY MATRIX**

<b>STATE</b> <b>COMMAND</b>	<b>BUILT-IN TEST</b>	<b>BUSY</b>	<b>DECLASSIFY</b>	<b>ERASE</b>	<b>ERROR</b>	<b>FAIL</b>	<b>FIND</b>	<b>IDLE</b>	<b>PLAY</b>	<b>POWER ON</b>	<b>RECORD</b>	<b>RECORD &amp; PLAY</b>
.BIT					X	X		X				
.CRITICAL	1		1	1	1	1	1	1	1		1	1
.DECLASSIFY					X			X				
.DISMOUNT					2			2				
.DUB					X			X			X	
.ERASE					X			X				
.EVENT	3				3	3	3	3	3		3	3
.FILES	X				X	X	X	X	X		X	X
.FIND					X			X			X	
.HEALTH	X		X	X	X	X	X	X	X		X	X
.HELP	X		X	X	X	X	X	X	X		X	X
.LOOP					X			X			X	
.MEDIA	X				X	X	X	X	X		X	X
.MOUNT					2			2				
.PLAY					X			X			4	
.RECORD					X		4	X	4			
.REPLAY					X			X			X	
.RESET	X	X	X	X	X	X	X	X	X		X	X
.SETUP	1		1	1	1	1	1	1	1		1	1
.SHUTTLE					X			X			X	
.STATUS	X	X	X	X	X	X	X	X	X		X	X
.STOP							X		X		X	X
.TIME	1		1	1	1	1	1	1	1		1	1
.TMATS					X			X				

**Legend**

- X Always valid.
- 1 Query function always valid. Changing masks, setup, or time only valid in IDLE or ERROR.
- 2 MOUNT and DISMOUNT only valid if not mounted or dismounted, respectively.
- 3 Query always valid. Declaring always valid in record, but not recording is system-specific.
- 4 Simultaneous recording and playing is system-specific.



**TABLE 6-19. REQUIRED COMMANDS**

Command	Recorder Type		
	Tape	Solid State	Disk
.BIT	M	M	M
.CRITICAL	O	O	O
.DECLASSIFY	O	M	M
.DISMOUNT	O	O	O
.DUB	O	O	O
.ERASE	O	M	M
.EVENT	O	O	O
.FILES	O	O	O
.FIND	M	M	M
.HEALTH	O	O	O
.HELP	O	O	O
.LOOP	O	O	O
.MEDIA	M	M	M
.MOUNT	O	O	O
.PLAY	M	M	M
.RECORD	M	M	M
.REPLAY	O	O	O
.RESET	M	M	M
.SETUP	O	O	O
.SHUTTLE	O	O	O
.STATUS	M	M	M
.STOP	M	M	M
.TIME	O	O	O
.TMATS	O	O	O
Legend			
M = Mandatory		O = Optional	