DRAFT

GOES HDR
Binary Protocol Specification

V0.9a

02/05/2024

Prepared by

Microcom Design, Inc.
10948 Beaver Dam Road, Suite C
Hunt Valley, MD 21030

For

National Oceanic and Atmospheric Administration

National Environmental Satellite, Data, and Information Service

NOAA/NESDIS

Table of Contents

List of Figures

List of Tables

1 Introduction

The purpose of this document is to extend the current GOES DCPRS Certification Standard to define a binary transmission format. The protocol parameters that will be defined in this document are the following:

- Overall Binary Protocol Message Structure (Section 3)
- Message Header Parameters (Sections 3.1 and 3.2)
- Message Footer Parameters (Sections 3.3 and 3.4)
- Open Binary Message Format (Section 4.1)
- Compact Pseudo Binary Message Format (Section 4.2)
- Compact Numeric ASCII Message Format (Section 4.3)
- Compact SHEF Alphanumeric ASCII Message Format (Section 4.4)
- Compact Full (Printable) ASCII Message Format (Section 4.5)

2 Background

Since its inception, the GOES DCPRS certification standards have left open the possibility for binary message transmissions to be defined. Currently only ASCII and Pseudo Binary (which uses a subset of the ASCII character set) formats have been defined and are in use. The reason for this is that a binary standard for such communications has been left "To Be Determined" in the GOES DCPRS certification standards. This document will fully define the binary protocol for use on the GOES DCS. This document also details how the binary protocol specification will provide a mechanism to enable users to rapidly transition to binary messages, significantly reducing the message lengths and making better use of DCS resources.

Specifically, this document defines five types of binary message formats which address different use cases and data character sets. The first and simplest message format is the Open Binary message structure which does not have any restrictions or specified decoding structures. The second is the Compact Pseudo Binary message format which uses the Pseudo Binary character set (64 ASCII characters representing all combinations of 6-bit binary values, plus space and forward slash) and then *compacts* the message data to be sent over the satellite link. The third is the Compact Numeric ASCII message formats which consists of only 21 characters. The fourth message format is Compact SHEF Alphanumeric ASCII which allows 47 ASCII characters to be encoded in the message. The fifth and final message format is Compact Full ASCII that can encode the full printable ASCII character set, which consists of 98 characters. While the Compact Full ASCII message format can be used to encode any ASCII message, the Compact Numeric and Compact SHEF message formats provide significantly higher compaction (aka compression) ratios.

The structure portion of the protocol defines the individual fields that are used and/or required in any binary message. The format portion of the protocol specifically addresses the data fields in the binary protocol regardless of the message format. In addition to defining an open binary message format, this protocol defines four additional message formats that will allow existing DCS messages to be compacted at the transmitter and de-compacted at the receiver. It is expected that these compaction schemes will reduce the time and cost for GOES DCS users to transition to binary since the data delivered to their information processing systems will be in the same format currently being utilized.

2.1 Current DCS Message Structure

Figure 1 below shows the current DCS ASCII and Pseudo Binary message structure. Except for a difference in the value in the Flag Word, the same structure is utilized for both message types since Pseudo Binary data utilizes a 64-character subset of the full 128-character ASCII set. In both cases, the message data portion is terminated with the non-printable ASCII End of Transmission (EOT) character (hexadecimal value 0x04). Further, both of these message formats require that the most significant bit in each byte is an Odd Parity bit.

Carrier 0.5s 0.25s	Clock 1-0-1 1=180	FSS 15-Bits	GOES ID 32-Bits	Flag Word 8-Bits	ASCII/PB Data with Odd Parity	ASCII EOT	Encoder Flush 32-Bits

Figure 1: ASCII and Pseudo Binary DCS Message Structure

The use of the EOT character and the Odd Parity bit in the data fields means that the Data portion of the current DCS message structure ***cannot*** support all 256 binary byte values, and therefore cannot support a binary message protocol.

As such, a modified version of the DCS Message Structure is required to support a DCS Binary Message protocol.

3 Binary Message Structure

The main difference between the Binary protocol and the legacy ASCII and Pseudo Binary protocols is that the message structure of the Binary protocol carries its data payload with an embedded general data packet structure rather than a sequence of ASCII or Pseudo Binary characters. The Binary protocol therefore uses a packet length field rather than a message terminator character (e.g., the ASCII EOT). The packet structure for a 300 bps binary message is shown in Figure 2; Figure 3 shows the packet structure for a 1200 bps binary message. Each packet consists of a 14-Bit length field, a 10-Bit BCH field, the data payload, and a 16-Bit CRC field. In the 1200 bps message structure the data payload and its 16-Bit CRC can be extended with up to four sections to make a 1200 bps message with a data length of up to 128,000 bits.

Packet Length 14-Bit	BCH 10-Bit	Data Bytes (Max: 32,000 bits)	CRC 16-Bit

Figure 2: Binary 300 bps General Packet Structure

Packet Length 14-Bit	BCH 10-Bit	Data Bytes (Max: 32,000 bits)	CRC 16-Bit	…	Data Bytes (Max: 32,000 bits)	CRC 16-Bit

Figure 3: Binary 1200 bps General Packet Structure

The 16-Bit CRC at the end of the data field for both 300 bps and 1200 bps is a hash function that produces a checksum, which allows verification of the packet data. This error detection checksum must be implemented to provide confidence that the data was received and decoded without error.

The complete message structure for both 300 bps and 1200 bps messages starting from the carrier and defining all of the message fields is shown in Figure 4. Comparing Figure 4 to Figure 1 in the previous section it can be seen that the first five fields; Carrier, Clock, FSS, GOES ID, and Flag Word; are the same as the first five fields in the ASCII and Pseudo Binary message structure.

The differences begin with the Packet Length following the Flag Word and carry on out to the last field, the Encoder Flush, as indicated by the bolded fields in Figure 4.

Carrier 0.5s 0.25s	Clock 1-0-1 1=180	FSS 15-Bits	GOES ID 32-Bits	Flag Word 8-Bits	**Packet Length 14-Bits**	**BCH 10-Bits**	**Binary Data**	**CRC 16-Bits**	**Encoder Flush 16-Bits**

Figure 4: Binary Message Structure Single Packet

While the field size of the Flag Word does not change in the Binary Message structure, bit designations are added and/or updated to support the new Binary format(s).

3.1 Updated GOES HDR Flag Word

The 8-Bit GOES HDR flag word immediately follows the GOES ID (as in the standard message formats), but is extended by the specification as defined in Table 1. Note that the bit numbering convention is the same as in the Certification Standard (i.e. the least significant bit is designated as Bit 1 and the most significant bit is Bit 8). Only the previously reserved, but unused, bits 3 and 4 have been changed to address the binary protocol implementation and provide an extension to the message format type.

Table 1: GOES HDR Flag Byte

Bit(s)	Name	Description
1 LSB	Spare	Send as 0
2	UTC Time Sync B_{TS}	0 = No UTC Time Sync since last transmission. 1 = UTC Time Sync since last transmission. Only used for Self-Timed Messages.
5/4/3	Extended Message Type B_{EMT}	Combined with B_{MT} bits. See Table 2.
7/6	Message Type B_{MT}	00 = Reserved 01 = ASCII 10 = Binary 11 = Pseudo Binary
8 MSB	Parity P_O	Odd Parity

Previously, Bit 3 was defined as "Data Compression" and Bit 4 was defined as "New Coding"; Bit 5 was not utilized. Bits 3, 4, and 5 are now combined to define the extended message types as defined in Table 2.

While the Message Type and Extended Message Type fields (B_{MT} and B_{EMT}) are shown as separate fields and defined to be backward compatible with existing DCS receive systems as indicated in Table 2, future updates to DCS receive systems should consider these bits as a combined 5-bit message type field as shown in Table 3 to support future expansion for new message types.

Table 2: Extended Message Type Definition

B_{MT}	B_{EMT}	Name	Description
00	000 … 111	Reserved	Possible Future Use
01	000	Standard ASCII	Standard or Legacy ASCII Format
01	001 … 111	Reserved	Possible Future Use
10	000	Open Binary	Open Binary Format (No Compaction)
10	001	Compact PB	Binary Format – Pseudo Binary Compaction
10	010	Compact NA	Binary Format – ASCII Numeric Compaction
10	011	Compact SA	Binary Format – SHEF ASCII Alphanumeric Compaction
10	100	Compact FA	Binary Format – Full ASCII Compaction
10	101 … 111	Reserved	Possible Future Use
11	000	Standard PB	Standard or Legacy Pseudo Binary Format
11	001 … 111	Reserved	Possible Future Use

Table 3: 5-Bit Combined Message Type Definition

B_{CMT}	Name	Description
00000 … 00111	Reserved	Possible Future Use
01000	Standard ASCII	Standard or Legacy ASCII Format
01001 … 01111	Reserved	Possible Future Use
10000	Open Binary	Open Binary Format (No Compaction)
10001	Compact PB	Binary Format – Pseudo Binary Compaction
10010	Compact NA	Binary Format – ASCII Numeric Compaction
10011	Compact SA	Binary Format – SHEF ASCII Alphanumeric Compaction
10100	Compact FA	Binary Format – Full ASCII Compaction
10101 … 10111	Reserved	Possible Future Use
11000	Standard PB	Standard or Legacy Pseudo Binary Format
11001 … 11111	Reserved	Possible Future Use

3.2 Packet Length and BCH

Following the 8-Bit Flag Word is the 14-Bit Packet Length field and the 10-Bit BCH field. The Packet Length field is defined as the number of bytes in the data field. Valid values for the message length field with 300 bps transmissions are from 0 to 4,000 bytes (32,000 bits) which per section 4.6.a of the Certification Standard is the maximum message length for 300 bps messages. Valid values for the message length field with 1200 bps transmissions are from 0 to 16,000 bytes (128,000 bits) which per section 4.6.a of the Certification Standard is the maximum message length for 1200 bps messages.

The 10-Bit BCH field following the Packet Length field uses the same Bose-Chaudhuri-Hocquenghem (31,21) (BCH) encoding scheme currently utilized in the DCP address. The lower 7-Bits of the Flag Word and the 14-Bits of the Message Length are combined to define the 21-Bit information portion of the code which is used to generate the 10-Bit check field.

The function of the Odd Parity bit in the Flag Word is unchanged and acts as a secondary layer of verification for the Flag Word. Specifically, Bit 8 of the Flag Word is set to either a 1 or a 0 based on Bit 0 through Bit 7 so there is an odd number of bits with a value of 1 when counting through all 8-bits of the Flag Word.

Figure 5 provides a graphical definition of which bits are included in the information and parity portions of the BCH encoding scheme.

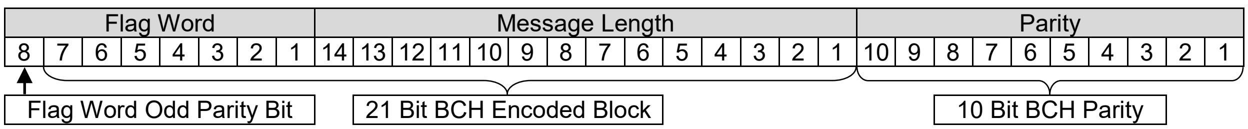

Figure 5: BCH Flag Word and Message Length Encoding Scheme

3.3 16-Bit CRC

The 16-Bit CRC has a polynomial generator defined as …

$$x^{16} + x^{15} + x^{13} + x^{9} + x^{7} + x^{6} + x^{5} + x^{3} + x + 1 \quad \text{(polynomial 0xd175).}$$

The CRC is generated from the preceding 0 to 4,000 bytes of packet data. For 1200 bps messages the data payload can exceed 4,000 bytes and when this occurs the first CRC will be inserted into the data payload and a second CRC will be started. This CRC will start with data byte number 4,001 and will continue up to 8,000 bytes. The maximum payload for a 1200 bps message is 16,000 bytes, which corresponds to a maximum of four CRC blocks.

Once generated, the 16 bit CRC shall be loaded into the transmit queue as two bytes in little endian format; i.e. the least significant byte shall be transmitted first followed the most significant byte.

It should be emphasized that the maximum number of *data* bytes is 16,000 (or 128,000 data bits), and that the CRCs are additional fields in the message structure. In other words, the CRCs do not reduce the DCS message data field size as defined in the current DCS Certification Standard (i.e. the *GOES Data Collection Platform Radio Set (DCPRS) CERTIFICATION STANDARDS, Version 2.0, June 2009*).

The addition of the CRC, Packet Length and BCH field do not violate the “Message Too Long” failsafe requirement of 110 seconds for either a 300 bps or a 1200 bps message. Specifically, the maximum 300 bps Binary message is 107.607 seconds in total transmission time, and the maximum transmit time for a 1200 bps Binary Message is 107.067 seconds inclusive of all of the message fields from carrier to flush.

3.4 Encoder Flush

At the end of the message after the final CRC 16-Bits have been loaded, zeros must be inserted into the message stream and appropriately scrambled to flush the encoder and decoder. As shown in Figure 4, the Encoder Flush field for the Binary Message structure is 16-Bits (i.e. 16 zeroes must be loaded following the final CRC value).

This differs from the 32-Bits of flush defined in section 3.4 of the June 2009 Version 2.0 Certification Standard since having a defined length allows a slightly shorter flush field as when using a termination value (i.e. EOT). In other words, this shortened Encoder Flush only applies to the Binary Message structure, the legacy ASCII and Pseudo Binary Message structure must still include the 32-bit flush.

4 Binary Message Formats

In addition to defining the binary message structure, this specification details the formats for the following five message types:

- Open Binary
- Compact Pseudo Binary
- Compact Numeric ASCII
- Compact SHEF Alphanumeric ASCII
- Compact Full ASCII

The Open Binary message format places no restrictions on the Data field in the Binary Message structure. This message format also does not require any additional processing beyond generating the Binary specific fields (i.e. the Packet Length, BCH Parity bits, and CRC-16 fields) defined in the Binary Message structure and detailed in the preceding sections.

The four "Compact" message formats listed above start with either Pseudo Binary or ASCII message data, and process it in such a way as to generate one of the four special Binary message formats: Compact Pseudo Binary, Compact Numeric ASCII, Compact SHEF Alphanumeric ASCII, or Compact Full ASCII.

Pseudo Binary to Compact Pseudo Binary provides a nearly one-to-one message correlation that only requires that the Pseudo Binary message to be compacted not include characters outside of the Pseudo Binary parameter representation set besides the two permissible special characters; slash (/) and space. Slash characters can be used to identify a reading not yet taken and/or invalid due to a sensor failure. Space characters are quite often used for message formatting, but are also sometimes used in place of the slash character.

For ASCII messages there are three options for the message compaction; two that use a reduced ASCII character set, and one that allows the full printable ASCII character set. If the original ASCII message data only includes numeric values and certain specific separators as defined in Table 4 and Table 5 below, the Compact Numeric ASCII can be utilized. If the original message also includes any upper-case letters (A through Z), such as in a SHEF coded ASCII message, but is still restricted to the characters defined in Table 6 below, then the Compact SHEF Alphanumeric ASCII can be utilized. Otherwise, the Compact Full ASCII compaction approach must be used.

For any of the four Compact message formats, the DCS transmitter (aka DCP) processes the Pseudo Binary or ASCII data and transmits the compacted binary data in the Binary Message structure. Upon reception the receiving equipment will de-compact the received message data back into its original Pseudo Binary or ASCII format. Note that while the information begins and ends in a non-compacted format, these messages are still considered binary message since the compacted bytes can take on any 8-bit value.

4.1 Open Binary Message Format

As noted above, the Open Binary message format does not impose any restrictions on the actual message data.

The Open Binary message structure is shown in Figure 6 below. Further, in Figure 6, the actual Flag Word value is also shown; the bit defined as X is the UTC Time Sync bit (B_{TS}), which is unrelated to message structure. The bit designated as P is the odd parity bit for the Flag Word.

Carrier 0.5s 0.25s	Clock 1-0-1 1=180	FSS 15-Bits	GOES ID 32-Bits	Flag Word								Packet Length 14-Bits	BCH 10-Bits	Data	CRC 16-Bits	Encoder Flush 16-Bits
				P	1	0	0	0	0	X	0					

Figure 6: Open Binary Message Structure

Note that the Flag Word byte defines a Message Type of Binary (B_{MT}=10) and the Extended Message Type bits designate no compaction; i.e. “Open Binary” (B_{EMT}=000).

Within each message, the 14-Bit Packet Length is defined as the number of bytes in the data field. In other words, the Packet Length does not include the 16-Bit CRC nor does it include the 14-Bit Packet Length. Valid values for the Packet Length are from 0 to 4,000 bytes (32,000 bits) for 300 bps messages, and from 0 to 16,000 bytes (128,000 bits) for 1200 bps messages.

4.2 Compact Pseudo Binary Message Format

The Compact Pseudo Binary Message is used to send Pseudo Binary data in a compressed format.

4.2.1 Pseudo Binary Character Compaction

Pseudo Binary bytes are ASCII characters in the format shown in Figure 7. The Compact Pseudo Binary message format removes the two most significant bits since these bits do not carry any information.

$$P_O \; 1 \; B_5 \; B_4 \; B_3 \; B_2 \; B_1 \; B_0$$

Figure 7: Pseudo Binary Bit Map

The resultant six information bits are concatenated together with the next six information bits taken from subsequent Pseudo Binary characters and reformed into bytes to generate the binary information to be transmitted. For example, Figure 8 shows how 4 Pseudo Binary bytes can be compacted into 3 binary bytes.

$P_0 1 a_5 a_4 a_3 a_2 a_1 a_0$ $P_0 1 b_5 b_4 b_3 b_2 b_1 b_0$ $P_0 1 c_5 c_4 c_3 c_2 c_1 c_0$ $P_0 1 d_5 d_4 d_3 d_2 d_1 d_0$

$a_5 a_4 a_3 a_2 a_1 a_0 b_5 b_4$ $b_3 b_2 b_1 b_0 c_5 c_4 c_3 c_2$ $c_1 c_0 d_5 d_4 d_3 d_2 d_1 d_0$

Figure 8: Example Pseudo Binary Compaction Process

4.2.2 Pseudo Binary Run Length Encoding

To allow the use of the space and slash characters (/) in a Pseudo Binary message, a modified form of run length encoding is used to distinguish between actual PB data and one or more consecutive space or slash characters. The slash character is used in standard PB messages to indicate a reading that has not yet been taken and/or a sensor that is not reporting properly. The space character can be used for message formatting and/or in place of the slash character in PB messages.

In typical run length encoding schemes, a repeat value is inserted ahead of each data value. If the data to be encoded has numerous long sequences of repeated characters, run length encoding can provide significant compression ratios. On the other hand, for data with little or no repeated character sequences, run length encoding can actually make the data size larger.

While repeated character sequences can and do happen in a PB message, it is not enough of a dominate characteristic for achieving significant compression ratios. Hence the need to modify standard run length encoding to adapt it to the type of data expected in a typical Pseudo Binary message, and the specific need to accommodate the slash character.

As noted in the previous section, the goal of the Compact Pseudo Binary standard is to reduce the overall data size by discarding the two non-informational bits in each byte to yield a series of 6-bit values that can then be concatenated together, and then reorganized into bytes for transmission. However, since the PB Character set includes 64 ASCII characters, of which slash is not one, the compacted PB data set includes all possible 6-bit combinations.

As such, there must be a way to include and identify space and slash characters in a Compacted PB message. Note that this was readily possible when dealing with 8-bit ASCII data, since the bit that is always a 1 in Figure 7 and Figure 8 above, is a 0 for the space and slash characters in the ASCII code set.

Distinguishing spaces and slashes from actual Pseudo Binary data is where the modified run length encoding indicators come in to the Compact Pseudo Binary specification. The definition of the three run length indicators is shown in Figure 9 through Figure 11.

4.2.2.1 Run Length Encoding Indicators

Since it is not expected to have long sequences of repeated PB characters in a typical DCS message, the Pseudo Binary (PB) Character run length encoding indicator shown in Figure 9 does *not* indicate that the next value should be repeated a number of times, but instead indicates the number of 6-bit PB values that follow. In other words, the number of consecutive PB characters that were compacted by the transmitter, and should be extracted by the receiver.

This 8-bit value always has the most significant bit equal to 1. The least significant 7 bits provide a count value of the number of compacted six-bit PB values that follow the indicator until the next run length indicator or until the end of the message. Since there is no reason to have a PB run length indicator unless PB Compacted values follow, a count of 0 is not needed so the binary 7-bit value is incremented by 1 to represent a count range from 1 to 128.

B7	B6	B5	B4	B3	B2	B1	B0
1	PB Character Count (1-128)						
PB Characters Encoding Identifier							

Figure 9: Pseudo Binary Character Run Length Encoding

The Space Run Length encoding indicator shown in Figure 10 is a 6-bit value with the two most significant bits (B5 and B4) always equal to 0. The least significant 4 bits provide a count value of the number of consecutive spaces in the input data, which is also the number of consecutive spaces to populate in the output data during the de-compaction process. The 4-bit Space Count is also incremented by 1 so a single 6-bit Slash Run Length indicator can represent from 1 to 16 consecutive spaces.

B5	B4	B3	B2	B1	B0
0	0	Space Count (1-16)			
Space Encoding Identifier					

Figure 10: Space Character Run Length Encoding

The Slash Run Length encoding indicator shown in Figure 11 has the same structure and function as the Space Run Length indicator shown above except that the next most significant bit (B4) is a 1 instead of a 0. A single Slash Run Length indicator can encode or decode 1 to 16 slash characters.

B5	B4	B3	B2	B1	B0
0	1	Slash Count (1-16)			
Slash Encoding Identifier					

Figure 11: Slash Character Run Length Encoding

4.2.2.2 Identifying, Distinguishing and Processing Run Length Indicators

As detailed in the previous section and shown in Figure 9 through Figure 11, the run length indicators (RLIs) have different bit sizes; i.e. 8 bits versus 6 bits. These indicators also utilize an entropy encoding structure to allow them to be identified and distinguished in the compacted data stream.

The first byte of the data block of a Compact Pseudo Binary message will be a RLI. During the de-compaction process, the first step is to identify the first RLI in the data stream by examining the first bit in the message data (i.e. the most significant bit in the first data byte). If this bit is a 1, then the first RLI is a Pseudo Binary Character indicator in the form of Figure 9. If the first bit is a 0, then the RLI is either a Space or Slash character RLI. To distinguish between a slash and a space, the next bit is examined; 0 indicates space while 1 indicates slash. Once the first RLI field is identified, the next group of bytes is processed accordingly.

In the case of a Pseudo Binary Character RLI, a count value is determined from the value of the next 7 bits and by adding one. Once the count value is calculated, the data stream is processed by extracting this number of consecutive 6-bit PB values. Each 6-bit value is then converted back to its equivalent 7-bit ASCII character, the odd parity is set accordingly, and this character appended to the output data bytes.

Following the last 6-bit PB value extracted per the calculated count value of the Pseudo Binary Character RLI, either the end of the compacted data or another RLI will occur. If it is not the end of the compacted data, then the next bit in the data stream determines the type of the next RLI. Note that next bit in the data stream is not necessarily the most significant bit of the next byte; instead, the bit to examine will depend on the number of 6-bit PB values extracted. In other words, the compacted data must be processed as a bit stream; not a byte stream.

For the case of a Space or Slash Character RLI, a count value is determined from the value of the next 4-bits plus one. Once this count value is determined, this number of ASCII spaces or slashes is simply appended to the output data bytes with the appropriate odd parity bit set in each character.

Following a Space or Slash Character RLI, another RLI will immediately follow unless the processing has reached the end of the message data. As noted above, the first bit immediately following the 6-bit Space or Slash RLI must be checked for the next RLI type. For example, if the first RLI field in the message is a Space or Slash Character type, then the next RLI will begin at bit 1 of the first byte of data in the message.

The sequence of identifying the next RLI, processing it, and extracting PB characters if it's an 8-bit RLI continues until the message data is exhausted and the binary CRC reached. Note that once processed, an RLI is simply discarded; i.e. the RLIs themselves are never included in output data bytes while they are used in calculating the CRC.

4.2.3 Compact Pseudo Binary Packet

Once compacted, the resulting binary bytes are packetized and transmitted. The message structure for a Compact Pseudo Binary message is shown in Figure 12. Note that the flag byte defines a Message Type of Binary (B_{MT}=10) and the Extended Message Type bits designate Compact Pseudo Binary (B_{EMT}=001).

Carrier 0.5s 0.25s	Clock 1-0-1 1=180	FSS 15-Bits	GOES ID 32-Bits	Flag Word								Packet Length 14-Bits	BCH 10-Bits	Data	CRC 16-Bits	Encoder Flush 16-Bits
				P	1	0	0	0	1	X	0					

Figure 12: Compact Pseudo Binary Message Structure

The Packet Length is the number of binary bytes being sent, this is the byte count after the data has been compacted and run length encoded.

4.3 Compact Numeric ASCII Message Format

The Compact Numeric ASCII Message format is used to compact ASCII messages that consist of 16 numeric ASCII characters (digits, polarity signs, decimal point, etc.). The base characters that can be represented in this format are shown in Table 4 along with the 4-bit binary code that designates the character in the compacted message. To create the data bytes for a message two four-bit codes are compacted together to form a single byte.

Table 4: Numeric ASCII Character Set

ASCII Character	Binary Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
Space	1010
+	1011
,	1100
-	1101
. (dp)	1110
/	1111

In addition to the base 16 characters, the Compact Numeric ASCII message format supports the special character translation of the two 4-bit codes shown in Table 5. Since the sequences shown in the Numeric Characters column are not encountered when representing numeric values, these special sequences may be utilized in this compaction scheme. During de-compaction to the numeric character set using Table 4, the receive system must also look for the sequences shown in Table 5, and translate these sequences back to the corresponding ASCII character(s).

Table 5: Numeric ASCII Special Character Translation

ASCII Character(s)	Numeric Characters	Binary Code
cr/lf	++	1011 1011
#	+-	1011 1101
=	-+	1101 1011
:	..	1110 1110
E	--	1101 1101

Any ASCII characters encountered during the compaction sequence not defined in Table 4 or Table 5 shall be replaced with the Space character prior to encoding. Similarly, if the resulting compacted message has an odd number of 4-bit codes, the 4-bit code for the ASCII space character must be used to complete the last byte.

Once a numeric ASCII message is compacted according to the rules above, the resulting binary bytes are packetized and transmitted. The message structure for the Compacted Numeric ASCII Message Format is shown in Figure 13. Note that the flag byte defines a Message Type of Binary (B_{MT}=10), and the Extended Message Type bits define Compact Numeric ASCII (B_{EMT}=010).

Carrier 0.5s 0.25s	Clock 1-0-1 1=180	FSS 15-Bits	GOES ID 32-Bits	Flag Word								Packet Length 14-Bits	BCH 10-Bits	Data	CRC 16-Bits	Encoder Flush 16-Bits
				P	1	0	0	1	0	X	0					

Figure 13: Compact Numeric ASCII Message Structure

The Packet Length is the number of binary bytes being sent; this is the byte count after the data has been compacted. Note that the special character translation sequences count as two 4-bit sequences even if the resulting ASCII equivalent is a single character.

4.4 Compact SHEF Alphanumeric ASCII Message Format

The Compact SHEF Alphanumeric ASCII Message format is used to compact (SHEF) ASCII messages that consist of the subset of the ASCII characters shown in Table 6. This compaction scheme focused on the common SHEF code ASCII messages used by many users and platforms. This compaction format only encodes the upper-case letters, and not any of the lower-case letters.

The first column of Table 6 is the numeric characters from Table 4. However, these characters are now encoded as a 5-bit binary value with the most significant bit being 0. An additional 31 characters, including the uppercase letters, are defined using a 6-bit code that has the most significant bit set to 1. This variable size code set provides a total of 47 characters with the six bit all ones code not assigned (N/A).

Table 6: Compact SHEF Alphanumeric ASCII Character Set

ASCII Character	Binary Code	ASCII Character	Binary Code	ASCII Character	Binary Code
0	00000	A	100000	Q	110000
1	00001	B	100001	R	110001
2	00010	C	100010	S	110010
3	00011	D	100011	T	110011
4	00100	E	100100	U	110100
5	00101	F	100101	V	110101
6	00110	G	100110	W	110110
7	00111	H	100111	X	110111
8	01000	I	101000	Y	111000
9	01001	J	101001	Z	111001
space	01010	K	101010	cr/lf	111010
+	01011	L	101011	#	111011
,	01100	M	101100	=	111100
-	01101	N	101101	:	111101
. (dp)	01110	O	101110	;	111110
/	01111	P	101111	N/A	111111

With the exception of the lower-case letters, any ASCII character encountered during the compaction sequence not defined in Table 6 shall be replaced with the Space character prior to encoding. It is permissible, but not required to convert the lower-case letters to upper case prior to encoding.

During the translation process, the 5 or 6-bit codes are continuously packed together to form bytes. Any unused bits in the last byte shall be set to 1. The compacted data is then packetized using the message structure shown in Figure 14, and then transmitted similarly to the other compaction schemes. The flag byte defines a Message Type of Binary (B_{MT}=10), and the compaction flag bits define Compact SHEF Alphanumeric ASCII (B_{EMT}=011).

Carrier 0.5s 0.25s	Clock 1-0-1 1=180	FSS 15-Bits	GOES ID 32-Bits	Flag Word								Packet Length 14-Bits	BCH 10-Bits	Data	CRC 16-Bits	Encoder Flush 16-Bits
				P	1	0	0	1	1	X	0					

Figure 14: Compact SHEF Alphanumeric ASCII Message Structure

As the data is received, the codes are de-compacted and reverse translated. The de-compaction algorithm first requires the examination of the next un-compacted bit to determine how many total bits to extract (0 => 5 bits or 1 => 6 bits).

The Packet Length is the number of binary bytes being sent; this is the byte count after the data has been compacted. To ensure any trailing bits are not erroneously decoded, unused bits must be filled with ones (1). Since the six bit all ones code in Table 6 has been reserved and an all ones 5-bit code is not defined, the trailing bits of 1's are discarded.

4.5 Compact Full ASCII Message Format

The Compact Full ASCII Message format is used to compact ASCII messages that consist of the Full printable ASCII character set. Table 7 shows the printable character encoding for this format. The first 32 printable characters beginning with space and ending with the question mark symbol (?) are encoded with 6 bits. The remaining 63 printable characters beginning with the at symbol (@) and ending with the tilde symbol (~) are encoded with 7 bits.

This variable size code set provides a total of 95 characters, which is equivalent to the full ASCII printable character set. The seven bit all ones code is both not assigned (N/A) and also supports the Special Code Translation shown in Table 8.

During the translation process, the 6 or 7-bit codes are continuously packed together to form bytes. Once all the characters are encoded and packed, any unused bits in the last byte are filles with ones. If the final 7 or fewer bits are all ones, no additional character is present.

This approach allows the seven all ones code to act as a special code translation indicator. It the 7-bit all ones sequence is encountered in the received data with a minimum of 2 more bits available, the next 2 bits define one of the three common ASCII control codes (HT, CR, LF) that can appear in text, and a special code that translates to CR and LF (CR/LF). Effectively, these codes become a 9-bit encoding sequence that then gets embedded in the compacted data when the ASCII codes are encountered. While it is not a requirement, it is recommended that before a CR is encoded, the next character be checked for a LF, and if it is the CR/LF code be utilized as this will provide the best compression.

The compacted data is then packetized using the message structure shown in Figure 15, and then transmitted similarly to the other compaction schemes. The flag byte defines a Message Type of Binary (B_{MT}=10), and the compaction flag bits define Compact Full ASCII (B_{EMT}=100).

Carrier 0.5s 0.25s	Clock 1-0-1 1=180	FSS 15-Bits	GOES ID 32-Bits	Flag Word								Packet Length 14-Bits	BCH 10-Bits	Data	CRC 16-Bits	Encoder Flush 16-Bits
				P	1	0	1	0	0	X	0					

Figure 15: Compact Full ASCII Message Structure

As the data is received, the codes are de-compacted and reverse translated. The de-compaction algorithm first requires the examination of the next un-compacted bit to determine how many total bits to extract (0 => 6 bits or 1 => 7 bits). As noted above, if the 7-bit all ones code is encountered, the next 2 bits, if available, must be extracted to get the appropriate ASCII control code(s).

Table 7: Compact Full (Printable) ASCII Character Set

ASCII Character	Binary Code	ASCII Character	Binary Code	ASCII Character	Binary Code
Space	000000	@	1000000	‘	1100000
!	000001	A	1000001	a	1100001
“	000010	B	1000010	b	1100010
#	000011	C	1000011	c	1100011
$	000100	D	1000100	d	1100100
%	000101	E	1000101	e	1100101
&	000110	F	1000110	f	1100110
‘	000111	G	1000111	g	1100111
(	001000	H	1001000	h	1101000
)	001001	I	1001001	i	1101001
*	001010	J	1001010	j	1101010
+	001011	K	1001011	k	1101011
,	001100	L	1001100	l	1101100
-	001101	M	1001101	m	1101101
.	001110	N	1001110	n	1101110
/	001111	O	1001111	o	1101111
0	010000	P	1010000	p	1110000
1	010001	Q	1010001	q	1110001
2	010010	R	1010010	r	1110010
3	010011	S	1010011	s	1110011
4	010100	T	1010100	t	1110100
5	010101	U	1010101	u	1110101
6	010110	V	1010110	v	1110110
7	010111	W	1010111	w	1110111
8	011000	X	1011000	x	1111000
9	011001	Y	1011001	y	1111001
:	011010	Z	1011010	z	1111010
;	011011	[	1011011	{	1111011
<	011100	\	1011100	|	1111100
=	011101	]	1011101	}	1111101
>	011110	^	1011110	~	1111110
?	011111	_	1011111	N/A or C/T	1111111

Table 8: Full ASCII Special Code Translation

ASCII Code(s)	Binary Code
HT	1111111 00
CR	1111111 01
LF	1111111 10
CR/LF	1111111 11

The Packet Length is the number of binary bytes being sent; this is the byte count after the data has been compacted. To ensure any trailing bits are not erroneously decoded, unused bits must be filled with ones (1). Since the seven bit all ones code in Table 7 has been reserved for special, an all ones 6-bit code is not defined, and since a byte consists of 8-bits, any trailing bits of seven or fewer 1's are to be discarded. Note that a final all ones byte is not valid unless some of the uppers bits are a continuation of the character that began in the next to last byte.

5 Binary Message Data Examples

This section will provide examples of the five Binary Message formats. The examples in this section will focus on the actual message data, and the fields bolded in Figure 16; i.e. Flag Word, Packet Length, BCH, Binary Data, and CRC.

Carrier 0.5s 0.25s	Clock 1-0-1 1=180	FSS 15-Bits	GOES ID 32-Bits	**Flag Word 8-Bits**	**Packet Length 14-Bits**	**BCH 10-Bits**	**Binary Data**	**CRC 16-Bits**	Encoder Flush 16-Bits

Figure 16: Binary Message Data Key Fields

5.1 Open Binary Message Data Example

Table 9 below shows a binary message example where all 256 byte values from 00 to FF in hexadecimal are transmitted sequentially. The total number of bytes transmitted after the GOES ID and before the Encoder Flush is 262; the 256 Data field values plus the 4 bytes encompassing the Flag Word, Packet Length and BCH check (first four bytes shaded in yellow), and the 2-byte CRC (last two bytes shaded in green).

Table 9: Open Binary Message Data Example

Byte Offset	Flag Word, Packet Length, BCH, Data and CRC Field Bytes															
0000	40	04	01	E7	00	01	02	03	04	05	06	07	08	09	0A	0B
0016	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
0032	1C	1D	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B
0048	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	3A	3B
0064	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49	4A	4B
0080	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B
0096	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B
0112	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	7A	7B
0128	7C	7D	7E	7F	80	81	82	83	84	85	86	87	88	89	8A	8B
0144	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B
0160	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB
0176	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB
0192	BC	BD	BE	BF	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB
0208	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB
0224	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB
0240	EC	ED	EE	EF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB
0256	FC	FD	FE	FF	55	4B										

5.1.1 Open Binary Example Flag Word

As shown above the Flag Word is hexadecimal **40** (or **01000000** in binary), which defines a Binary message with no compaction, i.e. an Open Binary message format. Note also that this value has an odd number of ones when written binary so the odd parity or most significant bit is 0.

5.1.2 Open Binary Example Packet Length and BCH Parity Check

The next three bytes (**04 01 E7**) or 24 bits are the Packet Length and BCH check fields. Written in binary the values are **00000100 00000001 11100111**.

Recollecting these bits into the 14-bit Packet Length yields …

00000100000000 = 0100 hexadecimal = 256 decimal.

The 10-bit BCH check field is …

0111100111 = 1E7 hexadecimal.

The 10-bit BCH field is generated using the standard BCH (31,21) algorithm (same as used to generate the GOES ID). The 21 information bits used to generate the 10-bit BCH parity check are the least significant 7 bits of the Flag Word combined with the 14-bit Packet Length; in this case …

100000000000100000000 = 100100 hexadecimal

5.1.3 Open Binary Example CRC-16

The CRC generated from the first data value of **00** through the last data value of **FF** is **4B55** hexadecimal. Note that since the CRC is sent in little endian format (i.e. least significant byte first, the equivalent byte values shown in Table 9 are reversed (**55 4B**).

5.2 Compact Pseudo Binary

5.2.1 Compact PB Message Data Example 1

To show a Compact Pseudo Binary example, it is first necessary to show a standard Pseudo Binary (PB) example, which is provided in Table 10 below. This example is a 153-byte PB message consisting of the single byte Flag Word and 152 bytes of PB data characters.

The data bytes in Table 10 have the Odd Parity bit intact at the most significant bit of each byte. The far-right column shows the 7-bit (i.e. parity bit stripped) ASCII equivalent characters for the data byte values to the left in each row.

Note that the Flag Word for this example is **E0** hexadecimal or **11100000** in binary, which designates this as a Pseudo Binary message. The parity bit is set to 1 since the lower seven bits have an even number of ones.

Table 10: Pseudo Binary Message Data Example 1

Byte Offset	Data Field Bytes																ASCII Characters
0000	**E0**	40	68	40	5E	40	7C	C1	46	C1	D0	C1	46	C2	D5	C2	`@h@^@|AFAPAFBUB
0016	5D	C2	57	C2	5B	C2	F2	C2	E0	40	54	40	4A	40	4A	40	]BWB[BrB`@T@J@J@
0032	54	40	5E	40	54	40	5E	40	5B	40	4A	40	D3	40	D0	40	]BWB[BrB`@T@J@J@
0048	4F	C2	C2	C1	F8	C2	EA	C2	FE	C4	C4	43	E6	C2	D0	C2	OBBAxBjB~DDCfBPB
0064	CD	C2	D9	C2	5B	C2	F7	C2	E6	40	F2	40	5E	40	54	40	MBYB[BwBf@r@^@T@
0080	F2	40	68	40	54	40	CE	40	D9	40	46	40	C7	40	4A	40	r@h@T@N@Y@F@G@J@
0096	C7	40	54	40	54	40	D3	40	54	40	D3	40	54	40	51	40	G@T@T@S@T@S@T@Q@
0112	51	40	51	40	51	40	51	40	51	C1	CB	C1	49	C1	49	C1	Q@Q@Q@Q@QAKAIAIA
0128	CB	C1	4C	C1	CE	BF	BF	BF	BF	BF	BF	BF	BF	BF	BF	BF	KALAN???????????
0144	BF	68	F4	40	5E	FE	F1	C4	49								?ht@^~qDI

Provided in Table 11 is the equivalent Compact Pseudo Binary message data for the same PB message shown in Table 10. First note that the message length has been significantly reduced and is now a total of 122 bytes. The message consists of 116 actual data bytes plus the 6 bytes of Binary message structure overhead.

5.2.1.1 Compact PB Example 1 Flag Word

As shown above the Flag Word is hexadecimal **C4** (or **11000100** in binary), which defines a Binary message with Pseudo Binary compaction, i.e. Compact Pseudo Binary message format. Note that the odd parity or most significant bit is 1 so that the complete byte has an odd number of bits equal to 1.

Table 11: Compact Pseudo Binary Message Data Example 1

Byte Offset	Flag Word, Packet Length, BCH, Data and CRC Field Bytes															
0000	C4	01	D1	AE	FF	02	80	1E	03	C0	46	05	00	46	09	50
0016	9D	09	70	9B	0B	20	A0	01	40	0A	00	A0	14	01	E0	14
0032	01	E0	1B	00	A0	13	01	00	0F	08	20	78	0A	A0	BE	10
0048	40	E6	09	00	8D	09	90	9B	0B	70	A6	03	20	1E	01	40
0064	32	02	80	14	00	E0	19	00	60	07	00	A0	07	01	40	14
0080	01	30	14	01	30	14	01	10	11	01	10	11	01	10	11	04
0096	B0	49	04	90	4B	97	04	C0	4E	FF	FF	FF	FF	FF	FF	FF
0112	FF	FF	A3	40	1E	FB	11	09	F9	F8						

5.2.1.2 Compact PB Example 1 Packet Length and BCH Parity Check

The next three bytes (**01 D1 AE**) or 24 bits are the Packet Length and BCH check fields. Written in binary the values are **00000001 11010001 10101110**.

Recollecting these bits into the 14-bit Packet Length yields …

00000001110100 = 0074 hexadecimal = 116 decimal.

The 10-bit BCH check field is …

0110101110 = 1AE hexadecimal.

The 10-bit BCH field is generated using the standard BCH (31,21) algorithm (same as used to generate the GOES ID). The 21 information bits used to generate the 10-bit BCH parity check are the least significant 7 bits of the Flag Word combined with the 14-bit Packet Length; in this case …

100010000000001110111 = 110077 hexadecimal

5.2.1.3 Compact PB Example 1 CRC-16

The CRC generated from the first data value of **FF** through the last data value of **09** is **F8F9** hexadecimal. Note that since the CRC is sent in little endian format (i.e. least significant byte first, the equivalent byte values shown in Table 11 are reversed (**F9 F8**).

5.2.1.4 Compact PB Example 1 Run Length Indicators

The RLIs for the Compact PB message of Table 11 are highlighted in light blue. Since no spaces or slashes appear in this example, both RLIs are of the PB Character type (i.e. 8-bit with the most significant bit being a 1).

The first RLI is the hexadecimal value **FF**, which indicates that 128 (**0x7F**+1 = 127+1 = 128) 6-bit PB characters follow. Since (128*6)/8 = 96, there are exactly 96 bytes containing these 6-bit PB values.

Following these 96 bytes is the next RLI, which ends up being byte aligned due to the PB data filling an exact number of bytes; i.e. the next RLI is **97** hexadecimal indicating that there are 24 PB data fields to follow (**0x17**+1 = 23+1 = 24), which encompasses the remaining portion of the message ((24*6)/8 = 18).

Note that the original PB message consisted of 152 data bytes, which yields a total of 912 bits or 114 bytes of message data. Added to the message data are the two 8-bit or byte RLIs for a total of 116 bytes in the Compact PB message as was indicated in Section 5.2.1.2

5.2.2 Compact PB Message Data Example 2

As a second Compact Pseudo Binary example, the previous example's data is modified to include four sequences of spaces and slashes in the middle of the message as shown in Table 12 below. This example is still using a 153-byte PB message consisting of the single byte Flag Word and 152 bytes of PB data characters.

As with the previous example, this example data also has the Odd Parity bit intact at the most significant bit of each byte, and the ASCII data is shown in the far-right column. The row with Byte Offset equal to 0064 is where the slashes and spaces begin. Since this is still a Pseudo Binary message, the Flag Word remains as **E0** hexadecimal or **11100000** in binary for this second example.

Table 12: Pseudo Binary Message Data Example 2

Byte Offset	Data Field Bytes																ASCII Characters
0000	E0	40	68	40	5E	40	7C	C1	46	C1	D0	C1	46	C2	D5	C2	`@h@^@|AFAPAFBUB
0016	5D	C2	57	C2	5B	C2	F2	C2	E0	40	54	40	4A	40	4A	40	]BWB[BrB`@T@J@J@
0032	54	40	5E	40	54	40	5E	40	5B	40	4A	40	D3	40	D0	40	]BWB[BrB`@T@J@J@
0048	4F	C2	C2	C1	F8	C2	EA	C2	FE	C4	C4	43	E6	C2	D0	C2	OBBAxBjB~DDCfBPB
0064	CD	2F	2F	2F	2F	20	20	20	20	2F	2F	2F	2F	20	20	20	M//// ////
0080	20	40	68	40	54	40	CE	40	D9	40	46	40	C7	40	4A	40	@h@T@N@Y@F@G@J@
0096	C7	40	54	40	54	40	D3	40	54	40	D3	40	54	40	51	40	G@T@T@S@T@S@T@Q@
0112	51	40	51	40	51	40	51	40	51	C1	CB	C1	49	C1	49	C1	Q@Q@Q@Q@QAKAIAIA
0128	CB	C1	4C	C1	CE	BF	BF	BF	BF	BF	BF	BF	BF	BF	BF	BF	KALAN???????????
0144	BF	68	F4	40	5E	FE	F1	C4	49								?ht@^~qDI

Provided in Table 13 is the equivalent Compact Pseudo Binary message data for the PB message shown in Table 12. The message length has been even further reduced and is now a total of 113 bytes. The message consists of 107 actual data bytes plus the 6 bytes of Binary message structure overhead.

5.2.2.1 Compact PB Example 2 Flag Word

As shown below the Flag Word is hexadecimal **C4** (or **11000100** in binary), which defines a Binary message with Pseudo Binary compaction, i.e. Compact Pseudo Binary message format. Note that the odd parity or most significant bit is 1 so that the complete byte has an odd number of bits equal to 1.

Table 13: Compact Pseudo Binary Message Data Example 2

Byte Offset	Flag Word, Packet Length, BCH, Data and CRC Field Bytes															
0000	C4	01	AE	85	BF	02	80	1E	03	C0	46	05	00	46	09	50
0016	9D	09	70	9B	0B	20	A0	01	40	0A	00	A0	14	01	E0	14
0032	01	E0	1B	00	A0	13	01	00	0F	08	20	78	0A	A0	BE	10
0048	40	E6	09	00	8D	4C	34	C3	C7	02	80	14	00	E0	19	00
0064	60	07	00	A0	07	01	40	14	01	30	14	01	30	14	01	10
0080	11	01	10	11	01	10	11	04	B0	49	04	90	4B	04	C0	4E
0096	FF	FF	FF	FF	FF	FF	FF	FF	FF	A3	40	1E	FB	11	09	7A
0112	8D															

5.2.2.2 Compact PB Example 2 Packet Length and BCH Parity Check

The next three bytes (**01 AE 85**) or 24 bits are the Packet Length and BCH check fields. Written in binary the values are **00000001 10101110 10000101**.

Recollecting these bits into the 14-bit Packet Length yields …

00000001101011 = 006B hexadecimal = 107 decimal.

The 10-bit BCH check field is …

1010000101 = 285 hexadecimal.

The 10-bit BCH field is generated using the standard BCH (31,21) algorithm (same as used to generate the GOES ID). The 21 information bits used to generate the 10-bit BCH parity check are the least significant 7 bits of the Flag Word combined with the 14-bit Packet Length; in this case …

100010000000001101011 = 11006B hexadecimal

5.2.2.3 Compact PB Example 2 CRC-16

The CRC generated from the first data value of **BF** through the last data value of **09** is **8D7A** hexadecimal. Note that since the CRC is sent in little endian format (i.e. least significant byte first, the equivalent byte values shown in Table 11 are reversed (**7A 8D**).

5.2.2.4 Compact PB Example 2 Run Length Indicators

The RLIs for the Compact PB message of Table 13 are highlighted in light blue. The first RLI is the hexadecimal value **`BF (10111111 binary)`**, which indicates that 64 (**`0x3F`**+1 = 63+1 = 64) 6-bit PB characters follow. Since (64*6)/8 = 48, there are exactly 48 bytes containing these 6-bit PB values.

Following these 48 bytes are the next RLIs, which encompass four consecutive bytes that mark the start of the slash and space sequences. Embedded in the first three hexadecimal bytes (**`4C 34 C3`**) are four 6-bit run length indicators as shown below:

```
     4C        34        C3
  01001100  00110100  11000011
010011  000011  010011  000011
  13      03      13      03
```

The first and third of these four is the slash RLIs with a repeat count of four (3+1), and the second and fourth are space RLIs, also with a repeat count of 4.

Following these 6-bit RLIs is the final RLI in this message; specifically, hexadecimal value **`C7`** indicating that there are 72 PB data fields to follow (**`0x47`**+1 = 71+1 = 72), which encompasses the remaining portion of the message.

Note that the original PB message consisted of 136 Pseudo Binary characters, 64 before and 72 after the slash and space sequences, which yields a total of 816 bits or 102 bytes of message data. Added to the message data are the two 8-bit RLIs and the four 6-bits RLIs for a total of 848 bits (816+2*16+4*6) or 107 bytes in the message as was indicated in Section 5.2.1.2.

5.3 Compact Numeric ASCII Message Example

To show a Compact Numeric ASCII example, it is first necessary to show a standard ASCII example that only includes characters from the subset defined by Table 4 and Table 5. This example is shown in Table 14 below. This example is a 318-byte ASCII message consisting of the single byte Flag Word and 317 bytes of ASCII data characters. The data bytes in the table also have the Odd Parity bit intact at the most significant bit of each byte. The far-right column shows the 7-bit (i.e. parity pit stripped) ASCII equivalent characters for the data byte values to the left in each row.

Note that in addition to digits and punctuation characters, the message also includes spaces (`20`), carriage returns (`0D`), and line feeds (**`0A`** or **`8A`** with the parity bit set); all of which are permissible.

The Flag Word for this example is **`20`** hexadecimal or **`00100000`** in binary, which designates this as an ASCII message. The parity bit is set to 0 since the lower seven bits have an odd number of ones.

Table 14: Numeric ASCII Message Data Example

Byte Offset	Data Field Bytes																ASCII Characters
0000	20	32	20	31	B3	BA	B3	B0	BA	B0	B0	20	B3	B0	B9	2C	2 13:30:00 309,
0016	B3	32	34	2C	34	AE	38	2C	B9	AE	31	2C	B3	AE	31	2C	324,4.8,9.1,3.1,
0032	B9	38	2C	37	B5	B9	AE	32	2C	B0	AE	B0	B0	2C	31	B6	98,759.2,0.00,16
0048	0D	8A	32	20	31	B3	BA	34	B0	BA	B0	B0	20	B3	31	32	..2 13:40:00 312
0064	2C	B3	34	32	2C	B3	AE	34	2C	B5	AE	B5	2C	B3	AE	B3	,342,3.4,5.5,3.3
0080	2C	B9	38	2C	37	B5	B9	AE	B3	2C	B0	AE	B0	B0	2C	34	,98,759.3,0.00,4
0096	B6	0D	8A	32	20	31	B3	BA	B5	B0	BA	B0	B0	20	B3	B3	6..2 13:50:00 33
0112	B6	2C	B3	B3	B0	2C	32	AE	B9	2C	34	AE	32	2C	B3	AE	6,330,2.9,4.2,3.
0128	B9	2C	B9	38	2C	37	B5	B9	AE	34	2C	B0	AE	B0	B0	2C	9,98,759.4,0.00,
0144	38	34	0D	8A	32	20	31	34	BA	B0	B0	BA	B0	B0	20	B3	84..2 14:00:00 3
0160	B5	31	2C	B3	B5	B5	2C	32	AE	31	2C	B3	AE	B3	2C	34	51,355,2.1,3.3,4
0176	AE	37	2C	B9	38	2C	37	B5	B9	AE	B5	2C	B0	AE	B0	B0	.7,98,759.5,0.00
0192	2C	31	32	31	0D	8A	34	20	31	34	BA	B0	B0	BA	B0	B0	,121..4 14:00:00
0208	20	31	32	AE	34	2C	31	31	AE	B3	0D	8A	32	20	31	34	12.4,11.3..2 14
0224	BA	31	B0	BA	B0	B0	20	B3	34	34	2C	B3	B6	B0	2C	31	:10:00 344,360,1
0240	AE	38	2C	B3	AE	37	2C	B5	AE	38	2C	B9	38	2C	37	B5	.8,3.7,5.8,98,75
0256	B9	AE	B6	2C	B0	AE	B0	B0	2C	31	B5	B5	0D	8A	32	20	9.6,0.00,155..2
0272	31	34	BA	32	B0	BA	B0	B0	20	B9	34	2C	B9	37	2C	B3	14:20:00 94,97,3
0288	AE	B0	2C	34	AE	B5	2C	B6	AE	B5	2C	B9	37	2C	37	B5	.0,4.5,6.5,97,75
0304	B9	AE	38	2C	B0	AE	B0	B0	2C	31	B9	B0	0D	8A			9.8,0.00,190..

Table 15: Compact Numeric ASCII Message Data Example

Byte Offset	Flag Word, Packet Length, BCH, Data and CRC Field Bytes															
0000	C8	02	99	5B	2A	13	EE	30	EE	00	A3	09	C3	24	C4	E8
0016	C9	E1	C3	E1	C9	8C	75	9E	2C	0E	00	C1	6B	B2	A1	3E
0032	E4	0E	E0	0A	31	2C	34	2C	3E	4C	5E	5C	3E	3C	98	C7
0048	59	E3	C0	E0	0C	46	BB	2A	13	EE	50	EE	00	A3	36	C3
0064	30	C2	E9	C4	E2	C3	E9	C9	8C	75	9E	4C	0E	00	C8	4B
0080	B2	A1	4E	E0	0E	E0	0A	35	1C	35	5C	2E	1C	3E	3C	4E
0096	7C	98	C7	59	E5	C0	E0	0C	12	1B	B4	A1	4E	E0	0E	E0
0112	0A	12	E4	C1	1E	3B	B2	A1	4E	E1	0E	E0	0A	34	4C	36
0128	0C	1E	8C	3E	7C	5E	8C	98	C7	59	E6	C0	E0	0C	15	5B
0144	B2	A1	4E	E2	0E	E0	0A	94	C9	7C	3E	0C	4E	5C	6E	5C
0160	97	C7	59	E8	C0	E0	0C	19	0B	BA	F8	4F				

Provided in Table 15 is the equivalent Compact Numeric ASCII message data for the ASCII message data shown in Table 14. First note that the message length has been significantly reduced and is now a total of 172 bytes (a compaction of 44.2%). The message consists of 166 actual data bytes plus the 6 bytes of Binary message structure overhead.

5.3.1 Compact Numeric ASCII Example Flag Word

As shown above the Flag Word is hexadecimal **C8** (or **11001000** in binary), which defines a Binary message with ASCII Numeric compaction, i.e. Compact Numeric ASCII message format. Note that the odd parity or most significant bit is 1 so that the complete byte has an odd number of bits equal to 1.

5.3.2 Compact Numeric ASCII Example Packet Length and BCH Parity Check

The next three bytes (**02 99 5B**) or 24 bits are the Packet Length and BCH check fields. Written in binary the values are **00000010 10011001 01011011**.

Recollecting these bits into the 14-bit Packet Length yields …

00000010100110 = 00A6 hexadecimal = 166 decimal.

The 10-bit BCH check field is …

0101011011 = 15B hexadecimal.

The 10-bit BCH field is generated using the standard BCH (31,21) algorithm (same as used to generate the GOES ID). The 21 information bits used to generate the 10-bit BCH parity check are the least significant 7 bits of the Flag Word combined with the 14-bit Packet Length; in this case …

100100000000010100110 = 1200A6 hexadecimal

5.3.3 Compact Numeric ASCII Example CRC-16

The CRC generated from the first data value of **2A** through the last data value of **BA** is **4FF8** hexadecimal. Note that since the CRC is sent in little endian format (i.e. least significant byte first, the equivalent byte values shown in Table 15 are reversed (**F8 4F**).

5.4 Compact SHEF Alphanumeric ASCII Message Example

To show a Compact SHEF Alphanumeric ASCII example, it is first necessary to show a standard ASCII example that only includes characters from the subset defined by Table 6. This example shown in Table 16 below is a typical DCS ASCII message that includes SHEF code designators (e.g. ":HG", ":VB", ":YB", etc.). This example is a 267-byte ASCII message consisting of the single byte Flag Word and 266 bytes of ASCII data characters. The data bytes in the table also have the Odd Parity bit intact at the most significant bit of each byte. The far-right column shows the 7-bit (i.e. parity bit stripped) ASCII equivalent characters for the data byte values to the left in each row.

Table 16: SHEF Alphanumeric ASCII Message Data Example

Byte Offset	Data Field Bytes																ASCII Characters
0000	20	BA	C8	C7	20	34	20	23	B5	20	31	B9	B0	AE	B0	B3	:HG 4 #5 190.03
0016	B9	20	31	B9	B0	AE	B0	B3	B9	20	31	B9	B0	AE	B0	B3	9 190.039 190.03
0032	B9	20	31	B9	B0	AE	B0	B3	B9	20	31	B9	B0	AE	B0	B3	9 190.039 190.03
0048	B9	20	31	B9	B0	AE	B0	B3	B9	20	31	B9	B0	AE	B0	B3	9 190.039 190.03
0064	B9	20	31	B9	B0	AE	B0	B3	B9	20	31	B9	B0	AE	B0	B3	9 190.039 190.03
0080	38	20	31	B9	B0	AE	B0	B3	38	20	31	B9	B0	AE	B0	B3	8 190.038 190.03
0096	B9	20	31	B9	B0	AE	B0	B3	B9	20	BA	D6	C2	20	B3	B9	9 190.039 :VB 39
0112	20	23	B6	B0	20	31	32	AE	B6	20	BA	D9	C2	20	B6	B0	#60 12.6 :YB 60
0128	20	23	B6	B0	20	31	32	AE	B0	20	BA	D9	C2	49	20	B3	#60 12.0 :YBI 3
0144	B9	20	23	B6	B0	20	B0	AE	B0	20	BA	D9	C2	54	20	B3	9 #60 0.0 :YBT 3
0160	B9	20	23	B6	B0	20	AD	B5	AE	32	20	BA	D9	46	20	B6	9 #60 -5.2 :YF 6
0176	B0	20	23	B6	B0	20	B3	B3	AE	B6	20	BA	D9	52	20	B6	0 #60 33.6 :YR 6
0192	B0	20	23	B6	B0	20	31	31	AE	B3	20	BA	D9	49	20	B3	0 #60 11.3 :YI 3
0208	B9	20	23	B6	B0	20	AD	34	AE	B9	20	BA	D9	D6	20	B3	9 #60 -4.9 :YV 3
0224	B9	20	23	B6	B0	20	31	32	AE	B6	20	BA	D9	D6	49	20	9 #60 12.6 :YVI
0240	B3	B9	20	23	B6	B0	20	B0	AE	B0	20	BA	54	57	20	B3	39 #60 0.0 :TW 3
0256	B9	20	23	B6	B0	20	B0	AE	B0	0D	8A						9 #60 0.0..

Note that in addition to digits, letters and punctuation characters; the message also includes spaces (**20**), carriage returns (**0D**), and line feeds (**0A** or **8A** with the parity bit set); all of which are permissible.

The Flag Word for this example is **20** hexadecimal or **00100000** in binary, which designates this as an ASCII message. The parity bit is set to 0 since the lower seven bits have an odd number of ones.

Provided in Table 17 below is the equivalent Compact SHEF Alphanumeric ASCII message data for the ASCII message shown in Table 16. First note that the message length has been significantly reduced and is now a total of 178 bytes (a compaction of 33.3%). The message consists of 172 actual data bytes plus the 6 bytes of Binary message structure overhead.

Table 17: Compact SHEF Alphanumeric ASCII Message Data Example

Byte Offset	Flag Word, Packet Length, BCH, Data and CRC Field Bytes															
0000	4C	02	B0	84	F6	79	94	45	76	55	05	20	70	06	95	05
0016	20	70	06	95	05	20	70	06	95	05	20	70	06	95	05	20
0032	70	06	95	05	20	70	06	95	05	20	70	06	95	05	20	70
0048	06	95	05	20	70	06	85	05	20	70	06	85	05	20	70	06
0064	95	05	20	70	06	95	7B	AC	2A	1A	55	D9	80	A0	89	C6
0080	57	BC	42	A3	01	5D	98	0A	08	9C	05	7B	C4	34	28	69
0096	57	66	02	80	E0	2B	DE	21	CD	43	4A	BB	30	14	D2	B8
0112	4A	F7	89	54	60	2B	B3	01	43	1B	8C	AF	78	C5	46	02
0128	BB	30	14	10	B8	6A	F7	8A	14	34	AB	B3	01	4D	23	92
0144	AF	78	D5	43	4A	BB	30	14	11	38	CA	F7	8D	68	50	D2
0160	AE	CC	05	01	C0	57	B9	EC	A1	A5	5D	98	0A	03	81	D7
0176	20	7E														

5.4.1 Compact SHEF Alphanumeric ASCII Example Flag Word

As shown above the Flag Word is hexadecimal **4C** (or **01001100** in binary), which defines a Binary message with SHEF ASCII Alphanumeric compaction, i.e. Compact SHEF Alphanumeric ASCII message format. Note that the odd parity or most significant bit is 0 so that the complete byte has an odd number of bits equal to 1.

5.4.2 Compact SHEF Alphanumeric ASCII Packet Length and BCH Parity Check

The next three bytes (**02 B0 84**) or 24 bits are the Packet Length and BCH check fields. Written in binary the values are **00000010 10110000 10000100**.

Recollecting these bits into the 14-bit Packet Length yields …

00000010101100 = 00AC hexadecimal = 172 decimal.

The 10-bit BCH check field is …

0010000100 = 084 hexadecimal.

The 10-bit BCH field is generated using the standard BCH (31,21) algorithm (same as used to generate the GOES ID). The 21 information bits used to generate the 10-bit BCH parity check are the least significant 7 bits of the Flag Word combined with the 14-bit Packet Length; in this case …

100110000000010101100 = 1300AC hexadecimal

5.4.3 Compact SHEF Alphanumeric ASCII Example CRC-16

The CRC generated from the first data value of **F6** through the last data value of **D7** is **7E20** hexadecimal. Note that since the CRC is sent in little endian format (i.e. least significant byte first, the equivalent byte values shown in Table 17 are reversed (**20 7E**).

5.5 Compact Full ASCII Message Example

To show a Compact Full ASCII example, the 307-byte DCS message shown in Table 18 was chosen. While this message may initially appear to be a Pseudo Binary message, closer inspection reveals the it includes ASCII characters outside of the Pseudo Binary character set; most notably the 0 and 1 digits in the top right cell, but also the numerous instances of the plus symbol (+). As such, the only compaction option for this message is the Compact Full ASCII message format.

Table 18: Full ASCII Message Data Example

<table>
<tr><th>Byte Offset</th><th colspan="16">Data Field Bytes</th><th>ASCII Characters</th></tr>
<tr><td>0000</td><td>20</td><td>43</td><td>C1</td><td>E6</td><td>F2</td><td>B0</td><td>31</td><td>B0</td><td>31</td><td>F8</td><td>F8</td><td>AD</td><td>40</td><td>C1</td><td>40</td><td>52</td><td>CAfr0101xx-@A@R</td></tr>
<tr><td>0016</td><td>DF</td><td>68</td><td>40</td><td>D5</td><td>40</td><td>46</td><td>40</td><td>40</td><td>40</td><td>40</td><td>40</td><td>DA</td><td>76</td><td>F1</td><td>C1</td><td>6E</td><td>_h@U@F@@@@@ZvqAn</td></tr>
<tr><td>0032</td><td>4C</td><td>C4</td><td>F1</td><td>46</td><td>F4</td><td>BF</td><td>62</td><td>BF</td><td>7A</td><td>E0</td><td>F7</td><td>52</td><td>FE</td><td>4C</td><td>43</td><td>6D</td><td>LDqFt?b?z`wR~LCm</td></tr>
<tr><td>0048</td><td>4C</td><td>C4</td><td>F1</td><td>AB</td><td>BF</td><td>FB</td><td>7A</td><td>BF</td><td>7C</td><td>DC</td><td>40</td><td>40</td><td>C1</td><td>C2</td><td>45</td><td>C2</td><td>LDq+?{z?|\@@ABEB</td></tr>
<tr><td>0064</td><td>45</td><td>C2</td><td>45</td><td>AB</td><td>BF</td><td>FB</td><td>D5</td><td>BF</td><td>7C</td><td>CE</td><td>BF</td><td>BF</td><td>FB</td><td>C2</td><td>40</td><td>C1</td><td>EBE+?{U?|N??{B@A</td></tr>
<tr><td>0080</td><td>BF</td><td>C1</td><td>FD</td><td>AB</td><td>BF</td><td>FB</td><td>7C</td><td>BF</td><td>7C</td><td>40</td><td>40</td><td>40</td><td>C4</td><td>C1</td><td>FE</td><td>C1</td><td>?A}+?{|?|@@@DA~A</td></tr>
<tr><td>0096</td><td>7C</td><td>C1</td><td>7A</td><td>AB</td><td>BF</td><td>7C</td><td>DA</td><td>BF</td><td>FB</td><td>EF</td><td>40</td><td>40</td><td>4F</td><td>C1</td><td>7A</td><td>C1</td><td>|Az+?|Z?{o@@OAzA</td></tr>
<tr><td>0112</td><td>79</td><td>C1</td><td>F7</td><td>AB</td><td>BF</td><td>7C</td><td>70</td><td>BF</td><td>7C</td><td>C7</td><td>BF</td><td>BF</td><td>FD</td><td>C1</td><td>F8</td><td>C1</td><td>yAw+?|p?|G??}AxA</td></tr>
<tr><td>0128</td><td>76</td><td>C1</td><td>F4</td><td>AB</td><td>BF</td><td>FD</td><td>7A</td><td>BF</td><td>FB</td><td>BF</td><td>40</td><td>40</td><td>46</td><td>C1</td><td>76</td><td>C1</td><td>vAt+?}z?{?@@FAvA</td></tr>
<tr><td>0144</td><td>75</td><td>C1</td><td>F2</td><td>AB</td><td>BF</td><td>FE</td><td>54</td><td>BF</td><td>7C</td><td>DC</td><td>40</td><td>40</td><td>C7</td><td>C1</td><td>F8</td><td>C1</td><td>uAr+?~T?|\@@GAxA</td></tr>
<tr><td>0160</td><td>76</td><td>C1</td><td>75</td><td>AB</td><td>BF</td><td>BF</td><td>D0</td><td>BF</td><td>FD</td><td>5D</td><td>40</td><td>40</td><td>C8</td><td>C2</td><td>45</td><td>C1</td><td>vAu+??P?}]@@HBEA</td></tr>
<tr><td>0176</td><td>7A</td><td>C1</td><td>FD</td><td>AB</td><td>BF</td><td>BF</td><td>73</td><td>BF</td><td>FE</td><td>E3</td><td>40</td><td>40</td><td>DC</td><td>C2</td><td>D9</td><td>C2</td><td>zA}+??s?~c@@\BYB</td></tr>
<tr><td>0192</td><td>58</td><td>C2</td><td>57</td><td>AB</td><td>BF</td><td>BF</td><td>5B</td><td>40</td><td>40</td><td>C4</td><td>40</td><td>40</td><td>68</td><td>C1</td><td>76</td><td>C2</td><td>XBW+??[@@D@@hAvB</td></tr>
<tr><td>0208</td><td>51</td><td>C2</td><td>43</td><td>AB</td><td>40</td><td>40</td><td>DC</td><td>BF</td><td>BF</td><td>FE</td><td>40</td><td>40</td><td>52</td><td>C1</td><td>43</td><td>C1</td><td>QBC+@@\??~@@RACA</td></tr>
<tr><td>0224</td><td>49</td><td>C1</td><td>CE</td><td>AB</td><td>BF</td><td>BF</td><td>F8</td><td>BF</td><td>BF</td><td>D0</td><td>40</td><td>40</td><td>C4</td><td>40</td><td>E6</td><td>40</td><td>IAN+??x??P@@D@f@</td></tr>
<tr><td>0240</td><td>67</td><td>40</td><td>70</td><td>AB</td><td>BF</td><td>BF</td><td>5B</td><td>BF</td><td>BF</td><td>C8</td><td>BF</td><td>BF</td><td>75</td><td>40</td><td>5D</td><td>40</td><td>g@p+??[??H??u@]@</td></tr>
<tr><td>0256</td><td>DA</td><td>40</td><td>61</td><td>AB</td><td>40</td><td>40</td><td>C7</td><td>BF</td><td>BF</td><td>FE</td><td>BF</td><td>BF</td><td>7A</td><td>40</td><td>58</td><td>40</td><td>Z@a+@@G??~??z@X@</td></tr>
<tr><td>0272</td><td>D6</td><td>40</td><td>D9</td><td>AB</td><td>BF</td><td>BF</td><td>62</td><td>BF</td><td>BF</td><td>CD</td><td>40</td><td>40</td><td>4F</td><td>40</td><td>D5</td><td>40</td><td>V@Y+??b??M@@O@U@</td></tr>
<tr><td>0288</td><td>54</td><td>40</td><td>D5</td><td>AB</td><td>BF</td><td>BF</td><td>52</td><td>40</td><td>40</td><td>C4</td><td>40</td><td>40</td><td>C7</td><td>40</td><td>54</td><td>40</td><td>T@U+??R@@D@@G@T@</td></tr>
<tr><td>0304</td><td>D3</td><td>40</td><td>54</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>S@T</td></tr>
</table>

The Flag Word (highlighted in yellow) for this example is `20` hexadecimal or `00100000` in binary, which designates this as an ASCII message. The parity bit is set to 0 since the lower seven bits have an odd number of ones.

Provided in Table 19 below is the equivalent Compact Full ASCII message data for the ASCII message shown in Table 16. First note that the message length has been reduced to a total of 265 bytes, which yields a compaction of 13.7% (significantly less than the other two ASCII compaction schemes provide, but with the tradeoff of being able to utilize the entire printable ASCII character set). The message consists of 259 actual data bytes plus the 6 bytes of Binary message structure overhead.

Table 19: Compact Full ASCII Message Data Example

Byte Offset	Flag Word, Packet Length, BCH, Data and CRC Field Bytes															
0000	D0	04	0F	D8	87	07	37	24	11	41	1F	1E	0D	81	06	05
0016	2B	FA	20	55	81	1A	04	08	10	20	5A	ED	C6	0E	E9	91
0032	38	C6	E8	FE	27	FD	60	EF	4B	F4	C8	7B	66	44	E2	5B
0048	FE	FD	3F	F2	E4	08	10	61	45	85	16	14	52	DF	F7	55
0064	FF	93	9F	7F	DC	28	10	5F	83	F4	B7	FD	FC	7F	E4	08
0080	10	22	41	FD	07	E4	1F	45	BF	F2	D3	FE	F7	C0	81	3E
0096	0F	A8	3E	60	F7	2D	FF	9C	1F	F9	1D	F7	FE	C1	F1	07
0112	B4	1E	85	BF	F7	D3	FE	DF	81	02	34	1E	D0	7A	C1	E4
0128	5B	FF	AA	3F	F2	E4	08	11	E0	F8	83	DA	0F	52	DF	7E
0144	83	FF	6E	C0	81	22	14	58	3E	A0	FD	2D	F7	F9	BF	FB
0160	1C	08	17	21	59	85	62	15	72	DF	7E	DC	08	11	20	40
0176	D1	07	B4	2A	30	A1	97	02	05	C7	DF	FD	02	05	28	30
0192	E0	C9	83	38	B7	DF	F0	FB	F4	20	40	89	03	34	0C	F0
0208	38	16	FB	F6	DF	7E	43	EF	F5	81	76	05	A8	18	4B	81
0224	02	3B	EF	FE	7D	FF	50	2C	40	AD	02	C9	6F	BF	89	F7
0240	E6	C0	81	3E	05	58	15	20	55	2D	F7	E9	40	81	12	04
0256	08	F0	2A	40	A7	02	A7	FA	7A							

5.5.1 Compact Full ASCII Example Flag Word

As shown above the Flag Word is hexadecimal **D0** (or **11010000** in binary), which defines a Binary message with Full ASCII compaction, i.e. the Compact Full ASCII message format. Note that the odd parity or most significant bit is 1 so that the complete byte has an odd number of bits equal to 1.

5.5.2 Compact Full ASCII Packet Length and BCH Parity Check

The next three bytes (**04 0F D8**) or 24 bits are the Packet Length and BCH check fields. Written in binary the values are **00000100 00001111 11011100**.

Recollecting these bits into the 14-bit Packet Length yields …

00000100000011 = 0103 hexadecimal = 259 decimal.

The 10-bit BCH check field is …

1111011100 = 3DC hexadecimal.

The 10-bit BCH field is generated using the standard BCH (31,21) algorithm (same as used to generate the GOES ID). The 21 information bits used to generate the 10-bit BCH parity check are the least significant 7 bits of the Flag Word combined with the 14-bit Packet Length; in this case …

101000000000100000011 = 140103 hexadecimal

5.5.3 Compact Full ASCII Example CRC-16

The CRC generated from the first data value of **87** through the last data value of **A7** is **7AFA** hexadecimal. Note that since the CRC is sent in little endian format (i.e. least significant byte first, the equivalent byte values shown in Table 19 are reversed (**FA 7A**).