DRAFT

GOES DCS
FHSS DCPC Specification

V0.2

03/08/2024

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

This document defines the proposed Frequency Hopping Spread Spectrum (FHSS) Data Collection Platform (DCP) Command (DCPC) communications protocol.

1.1 Frequency Hop Pattern and Reed Solomon Block

In order to better understand and present the communications protocol, it is instructive to first review the planned operational methodology of the FHSS DCPC link. As the description implies, the DCPC will utilize a frequency hopping approach that will spread the signal out across a wider spectrum. The spreading of the spectrum is necessitated by the National Telecommunications and Information Administration's (NTIA) Power Spectral Density (PSD) limits for the DCPC downlink from the National Oceanic and Atmospheric Administration's (NOAA) Geostationary Operation Environmental Satellite (GOES).

While the specifics of the NTIA's PSD limits are not germane to the communication protocol, the fact that DCPC uses frequency hopping is relevant. This relevance is due to the special timing relationship between the hop pattern sequence and the data transmission block since both are precisely synchronized to a Coordinated Universal Time (UTC) minute.

The frequency hopping uses a pattern that repeats every 60 hops. The hop dwell time is 0.1 seconds so the overall hop pattern repeats every 6 seconds. Moreover, during each hop 20 bits are transmitted making the overall data rate 200 bits per second (20/0.1 = 200).

The data transmission block consists of 250 bytes utilizing a Reed Solomon (RS) Forward Error Correction (FEC) block encoding scheme. Since 250 bytes equates to 2000 bits (8*250 = 2000), each RS block takes exactly 10 seconds to transmit. The sequencing of hop patterns and RS blocks, over a 60 second period, are shown below in Figure 1.

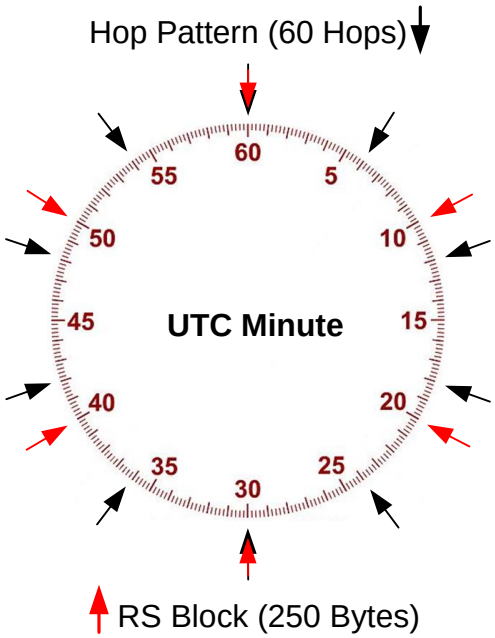

Figure 1: Hop Pattern and RS Block Alignment to Minute

In each minute there are 10 complete Hop Patterns and 6 complete RS Blocks both of which are precisely aligned to each UTC minute. Note also that a start of a hop pattern sequence and the start of an RS block only align at the top of the minute and the bottom of the minute. As such, it is necessary to synchronize to both of them before data can be received and properly processed.

If a receiver is already synchronized to UTC, obtaining hop and block alignment will be relatively straightforward. However, one of the goals of the DCPC link was to also allow it to provide a time synchronization source in much the same way that the predecessor signal, the DCP Interrogate (DCPI) did. As such, some of the characteristics and fields included in the communications are intended to provide this functionality as either a primary or secondary feature.

The following two subsections provide additional background details on the hop patterns and the Reed Solomon block that may provide useful insight as to the design and/or specifics of the communications protocol.

1.1.1 Frequency Hop Patterns

Two unique hop patterns have been defined for the FHSS DCPC system; one is to be used on GOES-East and the other is to be utilized on GOES-West. This allows for two independent data streams to command the DCPs in the field. Each pattern will use the same eight center frequencies, or bins, for the hops. The actual frequencies for the hops are not important so the discussion below will just use the generic references of F_1 through F_8.

The primary East sequence (S_E) is shown below, and starts with F_2 then stepping up through the even designators, followed by stepping down through the odd numbered

designations. This primary sequence occurs consecutively seven times for total of 56 hops. The final four hops use the shortened sequence shown at the end of E_{HP}.

$$S_E = F_2\ F_4\ F_6\ F_8\ F_7\ F_5\ F_3\ F_1 \qquad E_{HP} = S_E\ S_E\ S_E\ S_E\ S_E\ S_E\ S_E\ F_2\ F_4\ F_3\ F_1$$

The primary West sequence (S_W) is shown below, and starts with F_7 then stepping down through the odd designators, followed by stepping up through the even numbered designations. This primary sequence also occurs consecutively seven times for total of 56 hops. The final four hops use the shortened sequence shown at the end of W_{HP}.

$$S_W = F_7\ F_5\ F_3\ F_1\ F_2\ F_4\ F_6\ F_8 \qquad W_{HP} = S_W\ S_W\ S_W\ S_W\ S_W\ S_W\ S_W\ F_7\ F_5\ F_6\ F_8$$

Three things to note about these patterns are:

1. The patterns are mutually exclusive and never use the same frequency bin at the same time.
2. Moreover, while the patterns cross each other, at no time is one sequence hopping into a frequency bin previously occupied by the other sequence.
3. Each 4-bin shortened sequence is totally unique in that it never occurs in the other pattern, and it occurs in only one place in its pattern.

The first two items ensure the East and West downlinks will not interfere with each other.

Item 3 means the hop pattern itself can be used to identify the signal source, East versus West. It also means that each hop pattern provides a unique time alignment reference, i.e. each shortened sequence marks the end of the previous 60 hop pattern so it also identifies the start of the next full hop pattern. In other words, the end of the shortened sequence aligns with seconds 0, 6, 12, 18, 24, 30, 36, 42, 48, and 54 in the UTC minute.

1.1.2 Reed Solomon Block

The Reed Solomon Block is a special case of the standard RS(255,223) code (223 information bytes and 32 check bytes). To precisely align multiple blocks to the UTC minute, the block size has been reduced from 255 bytes down to 250 bytes by eliminating 5 information bytes. This creates a shortened or truncated Reed Solomon code designated as RS(250,218); i.e. there are 218 information bytes instead of 223 in the standard code, but the 32 check bytes remain.

When computing the check bytes and executing the Reed Solomon error correction algorithm, the standard RS(255,223) code is utilized with the last 5 information bytes set to 00 hexadecimal. In other words, these zero bytes are utilized in the Reed Solomon encoding and decoding process, but these bytes are omitted in the transmission of the block.

An interesting and useful side result of this approach is that these bytes serve as a mechanism to resolve the inherent phase ambiguity when initially locking to a continuous signal modulated with Binary Phase-Shift Keying (BPSK). Since the BPSK signal is continuous, it cannot contain a special carrier portion to obtain phase reference as is done with the short burst DCS transmissions sent by the DCPs. As such, the initial lock can be in one of two states; one state where the data is received as it should be (i.e. ones are 1’s and zeroes are 0’s) or the alternate state where all the data is received inverted (i.e. ones are 0’s and zeroes are 1’s).

The typical way to resolve this phase ambiguity is to include a special frame sync pattern at a known point in the block (typically at the start of the block). This approach takes bytes away from the information portion of the block, and serves no other useful purpose. However, for the FHSS DCPC communication protocol, a special frame sync is not required to resolve the BPSK phase ambiguity since it can be addressed with the truncated RS(250,218) code.

Since a RS(255,223) is transparent, if all the bytes are inverted, the code still checks. However, shortening or truncating the code eliminates the transparency. Specifically, if the RS(250,218) block is received inverted, then the zero fill bytes will be flagged as in error during the decoding process and the corrected value for all five bytes will be computed as FF hexadecimal (the inversion of 00 hex). When this occurs at reception, the receiver can be certain the data is inverted, and therefore enable the receiver to resolve the inherent BPSK phase ambiguity.

While a special fixed frame sync pattern is not required, the FHSS DCPC communication protocol does include a special 4 byte “Block ID” portion at the beginning of each block. The primary purpose of the Block ID portion is to combine with the frequency hopping pattern to provide a straightforward mechanism to align Reed Solomon blocks to the UTC minute. The secondary purpose of the Block ID is to provide synchronization to absolute UTC time. The Block ID is defined in Section 2.1.

However, the key feature of the way the hop pattern and Reed Solomon blocks are designed is two-fold: 1) if the time is known and accurate, synchronization can be achieved quickly; and 2) if time is not known, synchronization is still possible and an accurate time sync can be extracted from the FHSS DCPC signal.

2 FHSS DCPC Communication Protocol

The definition of the FHSS DCPC Communication Protocol begins with the mapping of the Reed Solomon block shown in Figure 2.

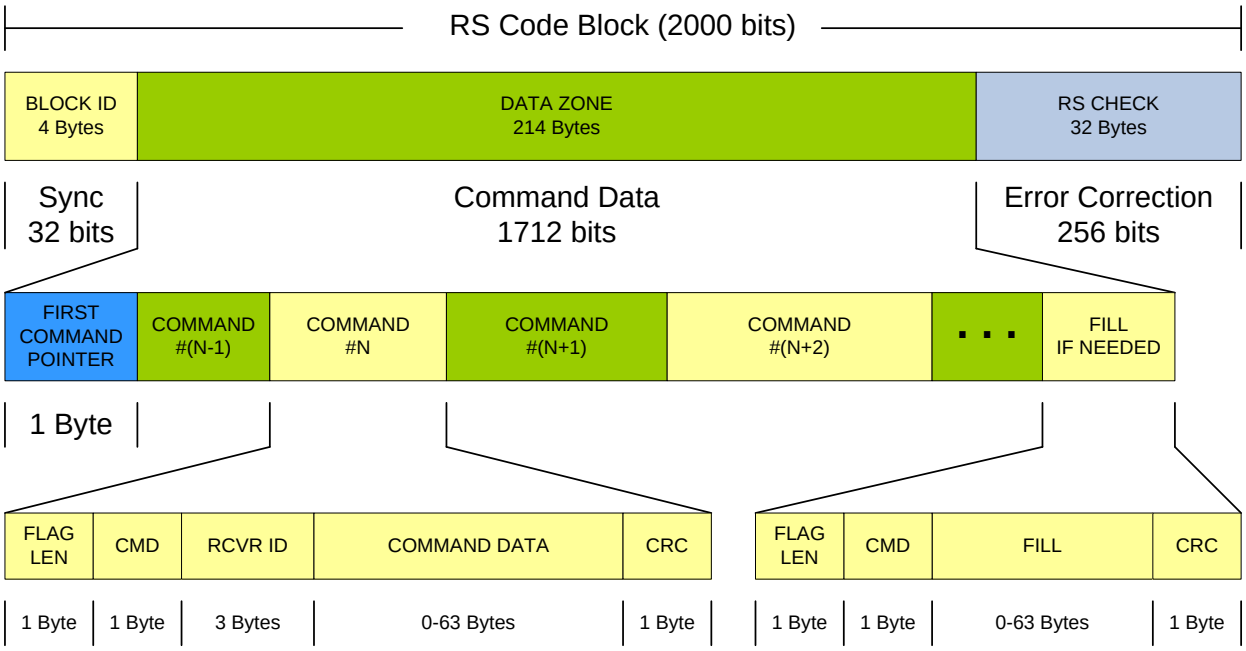

Figure 2: Reed Solomon Block Mapping

Provided below is a summary of the key features of the RS Code Block:

- RS Code Block:
  - Consists of 2000 bits or 250 bytes.
    - 218 information bytes.
    - 32 Reed Solomon error detection and correction bytes.
  - 10 seconds in duration; six RS Code Blocks per minute and synchronized to the UTC minute.
  - Each RS Code Block is divided into 3 fields: Block ID, Data, & Check
    - 4-byte Block ID field provides identification and timing information.
    - 214-byte Data field consists of command and/or fill packets.
    - 32-byte RS Check field allows for the correction of up to 16 erroneous bytes in the RS Block.
- Data Field:
  - Begins with the First Command Pointer (FCP) field, indicating the start location of the first new Command (or Fill packet) in the Data field.
    - Data bytes between the FCP and the first command are a continuation of the remaining bytes from the last command packet in the previous Frame.
    - The FCP allows command packets to cross Frame boundaries.

- If the First Command packet in the Frame is fully contained in the Frame, additional command packets immediately follow until the Frame is full or no more commands are in the transmit queue.
- Fill packets are added as needed to complete each frame; an entire frame could be Fill packets.

➢ Command Packets (6 – 69 bytes):

- Flag/Length field (1 byte):
  - The two most significant bits of this byte are sequence flags that are defined to support commands that span multiple Command Packets.
  - 11 => Complete; 01 => First; 00 => Continuation; 10 => Last
  - Complete (11) Command Packet does not span RS Code blocks.
  - Commands spanning more than one RS Code block begin with a First (01) Command Packet, possibly including one or more Continuation (00) Command Packets, and ending with a Last (10) Command Packet; all of these packets are then concatenated to form the complete Long command.
  - 6-Bit Length field specifies the length of the Command Data (e.g. 0-63 bytes).
- Command (1 byte):
  - Identifies the packet type
  - 8 Bits allows for up to 255 command codes plus the “Fill” packet type (00 hexadecimal).
  - Command before Receiver ID better supports Fill Packets.
- Receiver ID (3 bytes):
  - Receiver ID is different than the Platform ID.
  - “Fill” packet type sets this field to 000.
- Command/Fill Data (0-63 bytes):
  - User/Fill data field.
- CRC (1 byte):
  - Additional Command Packet validation.

The following subsections will detail the formatting and utilization of these fields.

2.1 Block ID (RS Code Block “BLOCK ID” Field)

The 4-byte Block ID field consists of a 1-byte bit-mapped flag field and a 3-byte sequence minute counter as shown in Table 1.

Table 1: Block ID Fields

Reception Order:	Byte 1	Byte 2	Byte 3	Byte 4
Significance:	N/A	Most	Middle	Least
Field:	Flag	Minute Counter		

2.1.1 Block ID Flag Field

The Block ID Flag byte is shown in Table 2 below. The two most significant bits (B6-B7) are a GOES Satellite confirmation identifier (East versus West). The next three most significant bits (B3-B5) are as yet undefined, and will be sent as 000. The three least significant bits (B0-B2) are the actual Block ID or order definition, but also serve an additional purpose as explained below.

Table 2: Block ID Flag Byte

B7	B6	B5	B4	B3	B2	B1	B0
					Block ID/Order: See Table 3		
		Format Specifier and/or Future Use – Send as 000					
Satellite Confirmation ID: 10 = East; 01 = West							

The 3-bit Block ID/Order field definition is provided in Table 3. Six of these values simply specify the order of the RS Block within the minute. However, when combined with the Minute Counter, the Block ID can be used to correlate the start and end of the block to absolute UTC. Further, since the start point of the hop pattern aligns with the start of Block 1 and Block 4, these points and the block information can be used to precisely set an internal clock.

The other two values are utilized to handle leap seconds that are required from time to time to keep UTC aligned with astronomical time. Generally, leap seconds are in the positive direction, but the possibility of a negative correction cannot be ruled out entirely so both types may need to be accounted for in the operation of the FHSS DCPC. However, in November 2022 the ITU at the 27TH Conference on Weights and Measures decided to abandon the leap second by or before 2035 so long-term the need to address leap seconds will go away.

Until the use of leap seconds is officially abandoned, some mechanism will need to be in place to address them when they do occur. Leap seconds are only added at the end of the year (12/31) or at the mid-year (6/30). The actual plan as to how to best address a leap second in the FHSS DCPC is still under consideration. What is known is that it will require a one-time and brief alteration of the hop pattern, and possibly the RS Block.

For example, for the positive leap second case, where the final 10 seconds of the day is extended by one second, the hopping at the end of the last 10 second RS Block could be suspended for one second. Alternately, for a negative leap second, where the final 10 seconds of the day is shortened by one second, it will most likely be necessary to suspend the DCPC transmission during these last nine seconds of the day, and then resume it at the beginning of the new day.

Table 3: Block ID/Order Field Definition

Bits	Value	Description/Meaning
000	0	Positive Leap Second
001	1	Block 1 – 00 to 10 Second
010	2	Block 2 – 10 to 20 Second
011	3	Block 3 – 20 to 30 Second
100	4	Block 4 – 30 to 40 Second
101	5	Block 5 – 40 to 50 Second
110	6	Block 6 – 50 to 00 Second
111	7	Negative Leap Second

For the negative leap second, the system will have to handle a 1 second shortening of the last 10 seconds of the day. This will be accomplished by suspending the hopping and the data transmission for the entire 9 seconds.

2.1.2 Minute Counter Field

The 3-byte Minute Counter is a minute-by-minute rolling value that is equal to the number of minutes since midnight January 1, 2024; the start of the last 4-year cycle that begins with a leap year. A 24-bit counter will reach a maximum value of $2^{24} - 1 = 16{,}777{,}215$ minutes, which is the equivalent of roughly 31.8 years.

The combination of the minute counter and the Block ID field effectively provides a block sequence counter, and when combined with the hopping pattern can be utilized to accurately set an internal clock to UTC.

2.2 First Command Pointer Field

The byte immediately following the Block ID field in the RS Code Block, is the First Command Pointer (FCP). This byte value provides an offset count of bytes into the Command Data where the next Command or Fill Packet starts.

The FCP allows Command/Fill Packets to cross the RS Code Block boundary. Specifically, a packet can start at or near the end of the previous RS Code Block, and complete at the beginning of the current RS Code Block. Referring to Figure 2, the FCP shall point to Command Packet #N. The section between Command Packet #N and the FCP is the completion of Command Packet #(N-1).

As Figure 2 also indicates, the completion of the previous packet may or may not be needed. If the FCP is equal to 1, the first Command/Fill Packet begins immediately following the FCP field, and there will not be a partial completion packet data.

The maximum Command Packet size is 69 bytes, this is also the maximum value of the FCP field. This is because the maximum partial completion packet data is 68 bytes, and the first Command/Fill Packet must begin 1 byte later. Therefore, the range of the FCP field is 1 to 69.

2.3 Command Packets

As noted above and indicated in Figure 2, each Command Packet consists of five fields. The following subsections detail these fields.

2.3.1 Flag/Length field (1byte)

The bit mapping of the Flag/Length composite field is shown in Table 4. The least significant 6 bits (B0-B5) define the Command Packet Data size in bytes (0-63); the total size of a Command is the Data size plus six bytes. The two most significant bits in the FLAG LEN byte field are the Packet Sequence Flags.

Table 4: FLAG/LEN Bit Map

B7	B6	B5	B4	B3	B2	B1	B0
		Packet Data Size					
Packet Sequence Flags							

The Packet Sequence Flags allows commands to be broken up into a series of standard sized Command Packets. The definition of these Packet Flags is summarized in Table 5.

Table 5: Packet Sequence Flags Definition

Bits	Value	Description/Meaning
00	0	Continuation Packet
01	1	Last Packet
10	2	First Packet
11	3	Compete Packet

Most commands are expected to be short and fit in one Command Packet. In these cases, the Packet Sequence Flags will be 11 indicating the complete command resides in just this one packet.

Long commands that will not fit in a single packet must be broken up into a series of packets beginning with a First (01) Command Packet, possibly include one or more Continuation (00) Command Packets, and end with a Last (10) Command Packet.

At the receiver, all of the received Command Data in the series of Command Packets are then concatenated to form the complete command. To ensure that all the necessary Command Data has been received, a packet sequence number consisting of a single byte will be included at the beginning of the Command Data field. Multiple packet commands are discussed further in Section 2.4.

2.3.2 Command Field (1 byte)

Following the Flags/Length byte is the Command field. The command field allows for up to 256 command codes. Two non-standard commands are the “fill” (00 hexadecimal) and “extended” (FF hexadecimal) which are defined in Section 4 along with other more standard commands. It is anticipated that as the FHSS DCPC evolves and matures, this

command set will be expanded as necessary until a final specification document is achieved.

2.3.3 Receiver ID Field (3 bytes)

A key concept of the proposed FHSS DCPC is that the Receiver ID is different than the Platform ID. Further, while the Platform ID is published on the DADDS website and known to all DCS uses, the corresponding Receiver IDs will not be published, and will only be known to individuals or agencies as needed to provide an initial layer of security.

Consideration is also being given to allowing one or two levels of group IDs to allow a single command to be sent to multiple DCPC capable platforms. However, to be beneficial such an option to allow a single command to be addressed to multiple systems would require that all the intended platforms are listening at the same time to be able to receive the command.

2.3.4 Command Data Field (0-63 bytes)

The Command Data bytes field is command dependent and will vary from 0 (empty) to up to 63 bytes. Long commands that require more than 63 data bytes will be broken up into a series of Command Packets as already discussed in Section 2.3.1, and will use the first byte of the Command Data Field as a packet sequence number. Multiple packet commands are discussed further in Section 2.5.

2.3.5 CRC Field (1 byte)

The 8-bit Cyclic Redundancy Check (CRC) byte is an error detection field for the Command Packet. The generator polynomial for the 8-bit CRC is provided below:

$$G(x) = x^8 + x^5 + x^4 + 1$$

The CRC is generated from all Command Packet fields: Flags/Length, Command byte, Receiver ID, and all Command Data bytes (if any).

2.4 Multiple Packet Commands

As noted above, if a command's data exceeds 63 bytes, the command must be broken up into multiple packets. When this occurs, the Command Packet Structure becomes that shown in Figure 3. Specifically, a Packet ID (PKT ID) byte immediately follows the RCVR ID field, and the maximum Command Data in the packet is reduced to 62 bytes. Also, each packet of a Multiple Packet Command must include at least one data byte.

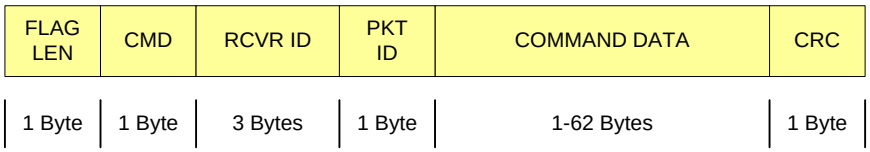

Figure 3: Multiple Packet Command Packet Structure

Since the Packet ID is a byte, a total of 256 Command Packets is supported making the maximum Command Data length 15,782 bytes (256*62). The packet with the First Packet Sequence Flag must have a Packet ID of 0x00. Subsequent Packet IDs must increment

by 1 for each successive Command Packet including the packet with the Last Packet Sequence Flag. Intermediate Command Packets between the First and Last, if any, must utilize the Continuation Packet Sequence Flag.

Note that each Command Packet in a Multiple Packet sequence must include the original Command Code, the same Receiver ID, and its own CRC check field.

It should also be noted, that the packets in a Multiple Packet Command may not, and for very long commands most likely will not, be sent consecutively. While the packets will always be sent in Packet ID order, other DCPC Command Packets can be interspersed in between to avoid tying up the DCPC Commanding system for an extended period of time.

Further, once a Multiple Packet Command has begun, the DCPC receiver must remain on, even if it uses a scheduled operation methodology (see Section 4.1.14), until the packet with the Last Packet Sequence Flag is received or a minimum of 15 minutes has elapsed since the last Packet Command was received for this command. If another packet for the command has not been received within this timeframe, the DCPC must assume that the Last Packet was corrupted and was missed, and the appropriate negative ACK sequence initiated (see Section 3.5).

2.5 Fill Packets

Due to the continuous nature of the FHSS DCPC, there will be times when the commands to be sent do not fill an entire RS Block Data field. Further, there will undoubtedly be times when no Command Packets are needed to be sent. During these periods, one or more Fill Packets will be required to fill out the RS Block.

2.5.1 Fill Data Bytes

The Fill Data bytes have no meaning and simply assist with “filling out” the unused portion of the RS Block’s Data Zone. However, the Fill Data will be a predetermined sequence of pseudo random bytes.

The specific pattern to be used is the last 63 bytes from the output value from the NIST randomness beacon from midnight January 1 2024, and is provided below. The fill packet will always be filled with the data below starting from byte 0xFB and proceeding to the right.

FBB08CDF380485AE61EF2FA4AD3B1298928927976AE2A8826B78A96F92241EA

2.6 Scrambler

The need for and definition of a Scrambler is still under consideration.

3 Command Acknowledgement

With the exception of a Fill Command or Packet, all DCPC commands addressed to and received by a platform must be acknowledged by the platform so that the system, NOAA, and the platform operator will know that the command was received, and either executed or rejected. In the case of a rejected command, the acknowledgement must indicate the primary reason the command could not be executed.

3.1 Acknowledgement Message Type

All acknowledgements must be sent by a Random Open/Standard Binary DCS message sent on one or more predetermined DCS channel(s). For complete details of the Open/Standard Binary DCS message format please refer to NOAA's *GOES HDR Binary Protocol Specification*.

Figure 4 shows the complete message structure for a binary message starting from the carrier and defining all of the required message fields. The actual DCPC command will be completely contained in the Binary Data field.

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: Standard Binary Message Structure

The 16-Bit CRC shown just after the Binary Data field is not the same as the 1-byte (8-bit) DCPC Command CRC, but is instead a separate hash function that produces a checksum, which allows verification of the message data.

3.2 Acknowledgment Reporting Methodology

DCPC Acknowledgements are a special case of the DCS Random Reporting, which means that it is possible for two separate Acknowledgements from different platforms to potentially collide with one another in time and neither be received. Accordingly, as with normal DCS Random reports and in accordance with NOAA's *Random Reporting User's Guide for the GOES DCS, July 15, 2021*, DCPC Acknowledgements shall be sent multiple times on a random interval using a uniformly distributed randomization algorithm.

While the specific characteristic of the Random Acknowlegment transmission is still under consideration, the current recommendation is for the following:

- Number of ACK Transmissions: 3
- ACK Random Interval: 5 minutes
- First ACK Transmission: One-half of the ACK Random Interval
- Randomization Percentage: 20%
- ACK Transmission Data Rate: 300 bps
- ACK Transmission Type: Standard Binary
- ACK Max Message Data: 74 bytes
- ACK Transmit Channel(s): TBD (minimum 1, maximum 3)

As the last bullet item indicates, the actual channel(s), and even the number of channels to utilize is still To Be Determined. Obviously, at least one DCS CS2 channel must be utilized, but it may be advantageous to use multiple channels to lower the potential of message corruption due to Radio Frequency Interference (RFI).

If multiple channels are used, the Random ACK transmission sequence shall always begin with the first defined channel, then proceed through the remaining defined channels sequentially, and then wrap back around to the first channel with a single transmission sent each time so that no two adjacent transmissions in time occur on the same channel. This process shall continue until all required transmissions are sent.

To avoid interfering with the sending of potentially mission critical data, Random ACK transmissions shall have lower precedence to Self-Timed transmissions and environmentally triggered Random transmissions. As such, DCP transmitters must include a scheduling algorithm to not impact these normal transmissions, but still strive to achieve the expected Random ACK interval while ensuring that all transmissions are sufficiently spaced out to not trip the DCP's Failsafe timer.

Naturally, once the initial DCPC Random Acknowledgement characteristics are finalized, this document will be updated to capture the final specifications. However, as covered in Sections 4.3.7 through 4.3.11, a compliant DCPC receiver must be designed to allow any, or all, of these parameters to be updated dynamically.

3.3 Common Command Acknowledge Codes

All DCPC commands of all forms must be acknowledged and must report at minimum a single byte Acknowledgement Code (ACK Code). While most DCPC commands will utilize command specific ACK Codes, Table 6 shows the Acknowledgment Codes common for all DCPC Commands. If a DCPC Command does have one or more command specific failure ACK Code(s), this information will be provided in the section detailing the specific DCPC Command.

Currently only the first several common ACK Codes are defined. The subsequent command code values are reserved and may be specified in a future revision of this specification.

Since it is possible that multiple errors with a DCPC command can occur, but only one ACK Code can be reported, the Error Priority column defined the order of precedence for the failure results. Note that the Command CRC Not Valid condition has the highest priority. A failed CRC identifies an error, but cannot pinpoint where the error occurs in the Command Packet; as such, an erroneous CRC trumps all other failures.

Note that it is possible that the RCVR ID is where the corruption occurred and the Command was actually intended for a different platform. Regardless, if the received RCVR ID matches the platform's ID, the platform must issue the Acknowledgment with ACK Code 0x04.

Table 6: Common Acknowledgement Codes

ACK CODE	Description/Meaning	Error Priority
0x00	Command Received and Successfully Executed	N/A
0x01	Unknown Command	2
0x02	Command Not Supported	2
0x03	Invalid Value in Command Data	3
0x04	Command CRC Not Valid	1
0x05	No Request/Status Command Form	4
0x06	Packets Missed in Multiple Packet Command	1
0x07	Reserved – Not Currently Defined	?
0x08	Reserved – Not Currently Defined	?
0x09	Reserved – Not Currently Defined	?

Also note that the Unknown Command and Command Not Supported have the same priority, but have different intended meanings. An Unknown Command indicates that this DCPC receiver does not recognize or know about this command; most likely since it was deployed before the command was added to the DCPC Protocol. A Command Not Supported means that the DCPC receiver knows the command exists, but it was an optional command and the platform does not support this functionality (e.g. the Hard Reset command defined in Section 0).

Finally, the ACK Code 0x05 is used when a command is received that is in a Request/Status form and the command does not implement such a response (e.g. see the Force GPS Fix Command in Section 4.1.15).

3.4 Single Command Packet Acknowledgment Format

The vast majority of FHSS DCPC commands will be contained in a single command packet and therefore will not exceed a total of 69 bytes. This section details the Acknowledgement Format for the single command packet Execute Commands and the single command packet Request/Status Commands.

Most of the commands detailed in Section 4 are Execute DCPC Commands, i.e. the command directs the platform to execute a function and/or change a configuration setting. When an Execute Command is issued it will typically contain Command Data. When a Single Packet Execute Command is received, the platform shall respond using the Single Packet Execute Command Acknowlegment Format as detailed in Section 3.4.1.

However, if the Execute Command is received without Command Data, it shall be treated as a Request Command; i.e. a request to return the existing configuration (e.g. current Self-Timed channel and data rate. When a Request Command is received, the platform must respond in the Single Packet Request/Status Command Acknowlegment Format as detailed in Section 3.4.2.

Some of the DCPC commands detailed in Section 4 are Status Requests and do not have an Execute form. Such commands shall also be acknowledged using the Single Packet Request/Status Command Acknowlegment Format.

3.4.1 Single Packet Execute Command Acknowledgement

The acknowledgment format for a single packet DCPC Execute Command must replicate the entire command packet and be followed by a single byte that indicates the command was successfully executed or the primary reason the command was rejected as shown in Figure 5. The entire acknowledgment shall be encapsulated in the Binary Data field of the Standard Binary Message as shown in Figure 4 above.

Figure 5: Single Packet Execute Command Acknowledgement Format

3.4.2 Single Packet Request/Status Command Acknowledgement

The acknowledgment format for a single packet DCPC Request/Status Command must first replicate the entire command packet. It then must be followed by the appropriate ACK Code based on the received command. If the ACK Code is 0x00, the requested status or configuration setting must be sent. If the command was rejected, is unknown, or is not implemented; the ACK Code must indicate the primary reason for the failure and no additional data provided. As such, the Request/Status Data field shown in Figure 6 is dashed to indicate that this data may or may not be present depending on the ACK Code. The entire acknowledgment shall be encapsulated in the Binary Data field of the Standard Binary Message as shown in Figure 4 above.

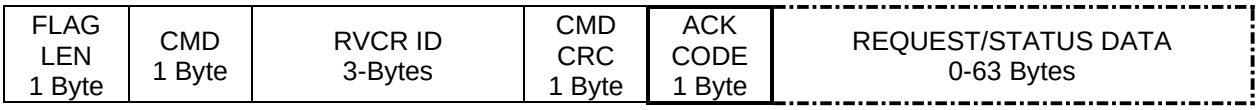

Figure 6: Single Packet Request/Status Command Acknowledgement Format

3.5 Multiple Packet Command Acknowledgment Methodology and Format

As covered in Section 2.4, the DCPC Protocol supports commands that can span many packets (up to 256) and bytes (up to 15,872). As such, a special Multiple Packet Command Acknowledgment, separate from the Single Packet Acknowledgement, is required. Three key distinctions between these two Acknowledgments are discussed below.

First, the Command Data is not included in the Acknowlegment due to the potential for the length of the data being quite large.

Second, if the Command Code is not recognized (i.e. unknown) or not supported, the negative ACK must be sent after receipt of the First Command Packet, and not after the entire command is received. This requirement is to allows the DCPC system to potentially cancel the transmission of as many as possible of the remaining unneeded Command Packets.

Third, if the command is recognized and supported, the Acknowledgment methodology must also handle the case where one or more Command Packets was either not received or received with an invalid CRC.

Finally, it should be noted that Multiple Packet Commands are only used for special case commands, which are all designated as Optional. As such, a DCPC does not have to fully support Multiple Packet Commands, but at a minimum must support being able to properly negative ACK that a command that could use multiple command packets is not supported.

3.5.1 Multiple Packet Command ACK All Packets Received

If a Multiple Packet Command is received in its entirety and without any CRC errors, the Acknowledgement format is that shown in Figure 7. Note that this format is similar to a single packet Request/Status ACK, but with the following key differences:

- The FLAG/LEN byte replaced with the number of packets received, which should be a value 1 more than the last Packet Sequence number received.
- The command is not echoed.
- There is not a CRC byte reported.
- The optional data field following the ACK Code is intended to provide result information that will be device and/or manufacturer specific.

As with all Acknowledgements, the ACK Code shall report whether or not the command was executed successfully or not.

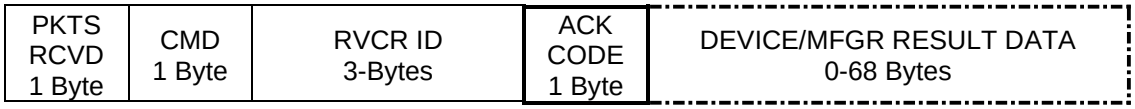

Figure 7: Multiple Packet Command All Received Acknowledgement Format

3.5.2 Multiple Packet Command Missed Packet(s) ACK

If a Multiple Packet Command is not received in its entirety, i.e. one or more packets was either not received at all or was received with a CRC error, the Acknowledgement format is that shown in Figure 8 shall be used. Note that ACK Code reported must be 0x06, and following the ACK Code a list of the Packet Numbers of the missed packets is sent to allow for retransmission of the missed packets. The first byte of the ACK is the count of good packets already received.

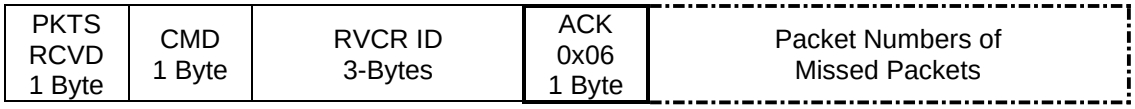

Figure 8: Multiple Packet Command Missed Packets Acknowledgement Format

While expected to be an extremely rare occurrence, it should be noted that it is possible that not all missed packets can be reported in a single Missed Packet ACK. Since it is recommended to utilize the Binary Message Type at 300 bps for Random ACKs, the maximum number amount of binary data is limited to 74 bytes to meet the maximum 3-

second duration of a 300 bps Random Transmission. As such, this limits the maximum size of the Missed Packet list to 68 bytes.

As discussed in the next section (3.5.3), the DCPC Multiple Packet Command approach supports missed packet retransmission with multiple retries to increase the probability of completing these special case commands. If the initial number of missed packets exceeds 68, the first 68 missed packets shall be reported in the first Random ACK sequence, and the remaining missed packets shall then be reported in following Random ACK sequence(s). As also detailed in Section 3.5.3, the number of retries that can be attempted to complete the reception of missed packets is limited; if exceeded, the command sequence must be aborted at both ends.

As noted in Section 2.4, once a Multiple Packet Command has begun, the DCPC receiver must remain on, even if it uses a scheduled operation methodology (see Section 4.1.14), until the packet with the Last Packet Sequence Flag is received or a minimum of 15 minutes has elapsed since the last Packet Command was received for this command.

If another packet for the command has not been received within this timeframe, the DCPC must assume that the Last Packet was corrupted and was also missed, and the ACK 0x06 sequence initiated. Further, the Packet Sequence number of 0xFF shall be added to the Missed Packet List, followed by the Packet Sequence number of the last good packet received to allow the DCPC system to determine all of the packets missed after the last packet that was successfully received.

After the Missed Packet ACK sequence has been initiated in the DCP transmitter, the DCPC receiver shall resume its normal DCPC Listen mode of operation, and wait for the DCPC system to begin the retransmit sequence. The already received packets must be saved, and be available to complete the execution of the command once all packets are received.

3.5.3 Multiple Packet Command Retransmission of Missed Packets

When the DCPC system begins the retransmission of missed packets, the DCPC system will proceed as follows depending on the number of packets that were missed.

If only one packet in the sequence was missed, the missed packet will be retransmitted with the Complete Packet Sequence Flag identifier, and the Packet Sequence number will match the previously utilized sequence number.

If two or more packets were missed, the first missed packet will be sent with the First Packet Sequence Flag identifier, and the Packet Sequence number will match the previously used sequence number; i.e. it is possible it will not be 0x00.

If only two packets were missed, the second missed packet will have the Last Packet Sequence Flag identifier, and again the Packet Sequence number will be the value of the second missed packet; i.e. it is possible the two Packet Sequence numbers will not be consecutive, but instead will match the two reported missed packets.

If three or more packets were missed, the second and subsequent missed packets up to but not including the last missed packet will utilize the Continuation Packet Sequence Flag identifier, and the final missed packet will have the Last Packet Sequence Flag identifier. All sent packets will use the previously sent, but missed, Packet Sequence

numbers. While the missed packets will be transmitted in numerical order, the packet numbers will most likely not be consecutive.

If all missed packets are received successfully this time, the command shall be immediately processed and executed, and the appropriate ACK sent.

If packets are again missed, the Missed Packet ACK sequence shall be executed, and the DCPC shall wait for a second retransmission sequence, which will proceed as the first transmission, but with only the remaining missed packets sent.

This process shall repeat until all packets are received, or five attempts, the first transmission plus four retransmits, have been made. If missed packets remain after the fifth attempt (fourth retransmission), the DCPC receiver and system shall cancel the command, and the DCPC system shall notify the DCS user that the command was unsuccessful.

4 Command Format

This section will detail the DCPC Command format for all currently specified commands. It should be noted that since a full byte value has been reserved for the DCPC Command, there are 255 possible commands excluding the Fill packet command code. However, not all possible command code values are currently defined, allowing for future expansion.

Recognizing the potential for ongoing future expansion of the DCPC commands, a DCPC compatible platform may have been deployed that supports an as yet to be defined command. As such, DCPC receivers must be designed to handle the case of an Unknown Command and return ACK Code 0x01 for any command code number the platform receives that it does not know how to process or execute.

Commands defined below are grouped by functionality. At the beginning of each functional group section, a table summarizing the commands will be provided. In each of the summary tables, the Command Byte code will be specified along with a short and long (aka verbose) description of the command.

The final column in each summary table will specify the “R/O” or required/optional use for each of the commands. Commands designated R must be implemented by any and all DCPC receivers and platforms. Commands designated O are optional to be implemented and supported. If a DCPC receiver/platform does not support an optional command, the ACK Code returned must be 0x02 or Command Not Supported.

In each command section heading, whether or not the command is required to be implemented or is optional is reiterated. Whereas in each command subsection, the intended utilization of the command is briefly discussed, and any required data for the Execute form of the command and/or the Request/Status form of the command will be specified. The command subsection will also detail the command specific error ACK Codes, if any, associated with the command. Command specific error codes will begin at 0x0A, and may not be unique to the command. In other words, the ACK Code value may be utilized in multiple commands and may have different meanings for each command.

Provided below is a table detailing and summarizing the meanings of the Type/Value Abbreviations used in the Command Specific tables throughout the command sections.

Table 7: Type/Value Abbreviations

Type/ Value	Description/Meaning
Byte Flag	Single Byte Either 0x00 or 0xFF only.
Byte ENUM	Single Byte Enumerated Value.
Byte BM	Single Bit-Mapped Byte
Byte UINT	Single Byte Unsigned Integer Value
Byte Char	Single ASCII Character
2-Byte ULE	2-Byte Unsigned Integer in Little Endian Order
2-Byte Time	2-Byte Time Value B1 Minutes, 0-59; B2 Seconds, 0-59.
3-Byte Time	3-Byte Time Value B1 Hours, 0-23[24]; B2 Minutes, 0-59; B3 Seconds, 0-59.
4-Byte ULE	4-Byte Unsigned Integer in Little Endian Order
4-Byte Float	4-Byte Floating Point Value in Little Endian Order
4-Byte D/T	Special version of a 4-Byte ULE where the value is a Date/Time computed as seconds since January 1, 2024 and is an Unsigned Integer in Little Endian Order
String	Multibyte ASCII Character String (NUL Terminated)

4.1 Control/Status Command Group

This section details the DCPC commands used for general control and status requests. Table 8 summarizes the commands that will be detailed in the following subsections.

Table 8: Control/Status Command Summary

CMD Code	Short Description	Long Description	R/O
0x00	Fill	Data has no meaning; it is used as a fill between commands	R
0x01	Ping	Requests a response from the platform to check if it is still active.	R
0x02	Software Reset	DCP must execute software reset after acknowledgement.	R
0x03	Hardware Reset	DCP should execute a hard reset after acknowledgement.	O
0x04	Disable Timed	Disable Self-Timed transmissions until specified date/time	R
0x05	Enable Timed	Enable Self-Timed transmissions (use after indefinite disable).	R
0x06	Disable Random	Disable Random transmissions until specified date/time	R
0x07	Enable Random	Enable Random transmissions (use after indefinite disable).	R
0x08	Enb/Dis DCP	Enable/Disable the DCP (if supported).	O
0x09	Failsafe Reset	Reset transmitter failsafe.	R
0x0A	Transmitter Status	Send DCP transmitter status and key performance metrics.	R
0x0B	Receiver Status	Send DCPC receiver status and key performance metrics.	R
0x0C	Set Platform ID	Set 32-Bit DCP Address	R
0x0D	Receiver Listen	Set DCPC receiver listen (aka power up) mode/times.	R
0x0E	Force GPS Sync	Force a GPS Sync and report result.	R
0x0F	Lat/Lon/TxID	Initiate a Lat/Lon/TxID Report Sequence	O
0x10	Resend Timed Tx	Resend a Self-Timed Message on Specified Channel	O
0x11	Future		?
thru	Future	**NOTE: Some of these could be system/manufacturer specific.**	?
0x1F	Future		?

4.1.1 Fill Command/Packet – Required

True Fill Packets must never be acknowledged by the DCPC receiver as these packets have no meaning and only serve to identify unused portions of each DCPC block and are therefore intended for all DCPC receivers. As shown in Figure 2 and detailed in Section 2.4, a true Fill Packet does not include a RCVR ID.

However, if an error occurs, it is possible that a DCPC receiver could receive a Command Packet with a Command Code value of 0x00. Normally, such an error would also be accompanied by a CRC error. For this occurrence, the ACK Code 0x04 (Command CRC Not Valid) must be sent. On the extremely remote possibility that a Command Code of 0x00 is received with a valid CRC, the platform must send an Acknowledgement with ACK Code 0x01 (Unknown Command).

4.1.2 Ping Command – Required

The Ping command is intended to verify DCPC communication to the platform. Upon receipt of a valid Ping command, the platform will simply provide and Acknowledgement response.

Table 9: Command Specifics – Ping Command

	Type/Value	Details/Notes
Command Code:	0x01	Request an ACK that the DCPC receiver is listening.
Command Data:	None	
Request Data:	None	
Special ACKs:	None	
Special Notes:	None	

4.1.3 Software Reset Command – Required

The Software Reset command must be supported by all platforms and requires that both the DCPC receiver and the DCP transmitter be able to execute a software reset; the ability to have the platform's datalogger also execute a software reset may be optionally supported. The Command Data for this command is a bit-mapped byte designating which component(s) is/are to be software reset.

Table 10: Command Specifics – Software Reset Command

	Type/Value	Details/Notes
Command Code:	0x02	
Command Data:	Byte BM	B2 = Logger; B1 = Receiver, B0 = Transmitter
Request Data:	None	
Special ACK:	0x1X	Cannot execute requested software reset.
Special Notes:	X indicates which components could not perform the reset.	

If the command can be executed, the Acknowledgement for this command must be sent after the software reset is completed (not before), to confirm that the Software Reset executed successfully.

If it is not possible for one or more of the requested components to execute a software reset, none of the units shall be reset and the negative ACK sent immediately. The Special ACK in this case is 0x10 with bits B2, B1, and B0 set according to which component or components that cannot be software reset at this time (B2 = Logger; B1 = Receiver, B0 = Transmitter).

Some examples of why a software reset cannot be performed are provided below.

- DCPC Receiver: Currently executing another command previously received.
- DCS Transmitter: DCPC receiver unable to communicate with the DCS transmitter.
- Datalogger: Platform does not support datalogger software reset.

4.1.4 Hardware Reset Command – Optional

The Hardware Reset command is an optional command to allow legacy platforms to add DCPC capability without requiring hardware changes. If supported, this command must at a minimum be able to hard reset both the DCPC receiver and the DCP transmitter; the ability to have the platform's datalogger also execute a hard reset may be optionally

supported. The Command Data for this command is a bit-mapped byte designating which component(s) is/are to be hard reset.

Table 11: Command Specifics – Hardware Reset Command

	Type/Value	Details/Notes
Command Code:	0x03	
Command Data:	Byte BM	B2 = Logger; B1 = Receiver, B0 = Transmitter
Request Data:	None	
Special ACK:	0x1X	Cannot execute requested hardware reset.
Special Notes:	X indicates which components could not perform the reset.	

Ideally, the hard reset to the DCP transmitter should be accomplished by a power cycle, but may optionally be accomplished by triggering a hard reset circuit. If the command can be executed, the Acknowledgement for this command must be sent after the hardware reset is completed (not before), to confirm that the Reset executed successfully.

If it is not possible for one or more of the requested components to execute a hardware reset, none of the units shall be reset and the negative ACK sent immediately. The Special ACK in this case is 0x10 with bits B2, B1, and B0 set according to which component or components that cannot be software reset at this time (B2 = Logger; B1 = Receiver, B0 = Transmitter).

Some examples of why a software reset cannot be performed are provided below.

- DCPC Receiver: Currently executing another command previously received.
- DCS Transmitter: DCS transmitter not powered up.
- Datalogger: Platform does not support datalogger software reset.

4.1.5 Disable Timed Command – Required

The Disable Timed command is used to disable GOES DCS Self-Timed transmissions until a specified date and time or indefinitely.

Table 12: Command Specifics – Disable Timed Command

	Type/Value	Details/Notes
Command Code:	0x04	
Command Data:	4-Byte D/T	Date/Time in seconds since January 1, 2024.
Request Data:	4-Byte D/T	Date/Time when Timed Tx's will be re-enabled.
Special ACK:	None	
Special Notes:	A value of 0x00000000 indicates indefinitely. A value of 0xFFFFFFF in Request Data indicates not Disabled	

4.1.6 Enable Timed Command – Required

The Enable Timed command is used to re-enable GOES DCS Self-Timed transmissions immediately; either after being disabled indefinitely or earlier than the previously specified date and time using the Disable Timed command.

Table 13: Command Specifics – Enable Timed Command

	Type/Value	Details/Notes
Command Code:	0x05	
Command Data:	None	
Request Data:	Byte Flag	Returns Timed enabled status: 0x00=Dis; 0xFF=Enb
Special ACK:	0x0A	Already enabled.
Special Notes:	None	

4.1.7 Disable Random Command – Required

The Disable Random command is used to disable GOES DCS Random transmissions until a specified date and time or indefinitely.

Table 14: Command Specifics – Disable Random Command

	Type/Value	Details/Notes
Command Code:	0x06	
Command Data:	4-Byte D/T	Date/Time in seconds since January 1, 2024.
Request Data:	4-Byte D/T	Date/Time when Random Tx's will be re-enabled.
Special ACK:	None	
Special Notes:	A value of 0x00000000 indicates indefinitely. A value of 0xFFFFFFF in Request indicates not Disabled.	

4.1.8 Enable Random Command – Required

The Enable Random command is used to re-enable GOES DCS Random transmissions immediately; either after being disabled indefinitely or earlier than the previously specified date and time using the Disable Random command.

Table 15: Command Specifics – Enable Random Command

	Type/Value	Details/Notes
Command Code:	0x07	
Command Data:	None	
Request Data:	Byte Flag	Returns Random enabled status: 0x00=Dis; 0xFF=Enb
Special ACK:	0x0A	Already enabled.
Special Notes:	None	

4.1.9 Enable/Disable DCP Command – Optional

This command is optional in the sense that the DCP transmitter must support an Enabled and Disabled state. If it is supported the DCPC receiver should be able to execute this command. In the Disabled state, no GOES DCS transmission shall occur. In the Enabled state, DCS transmissions will occur based on the current configuration of the DCP transmitter.

Table 16: Command Specifics – Enable/Disable DCP Command

	Type/Value	Details/Notes
Command Code:	0x08	
Command Data:	Byte Flag	0x00=Disable; 0xFF=Enable
Request Data:	Byte Flag	Enabled/Disabled status: 0x00=Dis; 0xFF=Enb
Special ACK 1:	0x0A	Already enabled/disabled.
Special ACK 2:	0x0B	Could not enable due to a configuration condition.
Special Notes:	Send 0x02 if the DCP transmitter does not use/support Enabled/Disabled states.	

4.1.10 Failsafe Reset Command – Required

This command is used to reset the Failsafe of the DCS transmitter.

Table 17: Command Specifics – Failsafe Reset Command

	Type/Value	Details/Notes
Command Code:	0x09	
Command Data:	None	
Request Data:	Byte Flag	Returns Failsafe Status: 0x00=OK; 0xFF=Tripped
Special ACK:	0x0A	Failsafe is not tripped.
Special Notes:	None	

4.1.11 Transmitter Status Command – Required

This is a status request only command that returns status and performance metrics.

Table 18: Command Specifics – Transmitter Status Command

	Type/Value	Details/Notes
Command Code:	0x0A	
Command Data:	None	Status Request Only
Status Data:	Byte Flag	Enabled/Disabled status: 0x00=Dis; 0xFF=Enb
	4-Byte D/T	Date/Time of Last Timed Tx (0x00000000=None)
	Byte	Last Timed Transmission Result (see Table 19)
	4-Byte D/T	Date/Time of Last Random Tx (0x00000000=None)
	Byte	Last Random Transmission Result (see Table 19)
	4-Byte D/T	Date/Time of Last GPS Synce (0x00000000=None)
	4-Byte D/T	Date/Time of Next Timed Tx (0x00000000=None)
	4-Byte D/T	Date/Time of Next Random Tx (0x00000000=None)
	Byte Flag	Returns Failsafe Status: 0x00=OK; 0xFF=Tripped
	Byte UINT	Power Supply Voltage *10 (ex: 123 =12.3V)
Special ACK:	0x0A	DCPC Receiver not able to communicate with DCP.
Special Notes:	Date/Time values in seconds since January 1, 2024.	

Table 19: Command Specifics – Last Transmission Result Codes

LAST TX CODE	Description/Meaning
0x00	Transmission Sent – Parameters Good
0x01	Transmission Sent – Low Forward RF Power
0x02	Transmission Sent – High VSWR (>2:1)
0x03	Transmission Fail – Failsafe Tripped
0x04	Transmission Fail – Low Battery Voltage
0x05	Transmission Fail – Frequency Error
0x06	Transmission Fail – Temperature Out of Range
0x07	Transmission Fail – DCP/Mfgr Specific
0x08	Transmission Fail – DCP/Mfgr Specific
0x09	Transmission Fail – DCP/Mfgr Specific

4.1.12 Receiver Status Command – Required

This is a status request only command that returns status and performance metrics. The actual information that this request will return is still under consideration. The parameters shown in the Table 20 are suggested metrics.

Table 20: Command Specifics – Receiver Status Command

	Type/Value	Details/Notes
Command Code:	0x0B	
Command Data:	None	Status Request Only
Status Data:	2-Byte ULE	Received Signal Level *-10 (ex: 1234 = -123.4 dBm)
	Byte	Last Command Code Received and for Platform
	Byte	ACK for Last Command Code Received for Platform
	Byte UINT	Power Supply Voltage *10 (ex: 123 =12.3V)
Special ACK:	0x0A	Not in Already enabled/disabled.
Special Notes:	None	

4.1.13 Set Platform ID Command – Required

This command is used to set, change, or request the platform's DCP Platform ID (aka NESDIS ID). Note that this is not the DCPC receiver's ID.

Table 21: Command Specifics – Platform ID Command

	Type/Value	Details/Notes
Command Code:	0x0C	
Command Data:	4-Byte ULE	New DCP Platform ID
Request Data:	4-Byte ULE	Current DCP Platform ID
Special ACK:	None	
Special Notes:	None	

4.1.14 DCPC Receiver Listen Command – Required

This command is used to set, change, or request the platform's DCPC Listen schedule to support various modes of operation to reduce power consumption. The data for this command always begins with a Mode byte and then may have additional values depending on the Mode as shown in Table 22.

Table 22: Command Specifics – Receiver Listen Command

	Type/Value	Details/Notes
Command Code:	0x0D	
Option 1		
Command Data:	0x00 (Byte)	Mode 0 – Always Listening – No additional data.
Option 2		
Command Data:	0x01	Mode 1 – Listen After Self-Timed Transmission (ST Tx)
	Byte UINT	Minutes, receiver will be listening following ST Tx.
Option 3		
Command Data:	0x02	Mode 2 – Listen At Specified Interval
	Byte UINT	Listen Interval in Hours (See Special Notes)
	2-Byte ULE	Listen Offset in Minutes (See Special Notes)
	Byte UINT	Minutes receiver will be listening each period.
Request Data:	Mode+Data	In same format based on Mode.
Special ACK:	0x0A	Mode not supported
Special Notes:	Interval must be evenly divisible into 24 (1, 2, 3, 4, 6, 8, 12, 24). Offset must be less than 60*Interval.	

4.1.15 Force GPS Sync Command – Required

The Force GPS Sync command directs the platform to execute an immediate GPS synchronization if a GPS receiver is in use by the platform. If a GPS receiver is used by the platform, then the Acknowledgement for this command must be sent after the GPS Sync unless an issue prevents the sync from starting.

Table 23: Command Specifics – Force GPS Sync Command

	Type/Value	Details/Notes
Command Code:	0x0E	
Command Data:	None	
Request Data:	None	
Special ACK 1:	0x0A	Cannot execute a GPS Sync at this time.
Special ACK 2:	0x0B	Platform syncs using DCPC downlink (i.e. not GPS).
Special Notes:	None	

4.1.16 Lat/Lon/TxID Command – Required

The command directs the platform initiate a Lat/Lon/TxID Identify Message sequence. The Acknowledgement for this command is immediate, and not after the Lat/Lon/TxID sequence is complete.

Table 24: Command Specifics – Lat/Lon/TxID Command

	Type/Value	Details/Notes
Command Code:	0x0F	
Command Data:	None	
Request Data:	None	
Special ACK:	0x0A	Cannot execute a Lat/Lon/TxID sequence.
Special Notes:	Send 0x02 if the DCP transmitter does not support Lat/Lon/TxID.	

For complete details of the Lat/Lon/TxID Identify Messaging requirement please refer to NOAA's *GOES HDR GOES HDR Latitude/Longitude/Transmitter Identification Specification*.

4.1.17 Resend Timed Tx Command – Optional

The Resend Timed Tx command allows for a mechanism to resend a previous Self-Timed message to be able to recover data that was missed due to corruption in the transmission's message data. This is an optional command since to be utilized the DCP transmitter must have a mechanism to capture and store prior Timed transmissions for retransmission. Further, in order for the command to be addressed, the transmitter must have sufficient memory to still have the requested transmission saved.

This command requires two parameters: 1) The Date and Time of the Self-Timed Transmission to be retransmitted referenced to the top of the Self-Timed Window; 2) the channel to resend the transmission on.

The retransmitted message must be sent on the channel supplied in the command and must be at the same data rate and in the same message format as it was originally sent.

A positive acknowledgement to this command must be sent after the last Self-Timed message is re-transmitted, while a negative acknowledgment should be sent upon receipt of the command.

Table 25: Command Specifics – Resend Last Timed Message Command

	Type/Value	Details/Notes
Command Code:	0x10	
Command Data:	4-Byte D/T	Date/Time of Requested Timed Transmission
	2-Byte ULE	Retransmit On Channel (1-266, 301-566)
Request Data:	None	
Special ACK:	0x0A	Channel not supported (possible future use).
	0x0B	Need a valid 1200 bps channel for retransmission.
	0x0C	No Self-Timed Transmission at specified Date/Time
	0x0D	No longer have requested transmission saved.
Special Notes:	Send ACK Code 0x00 after retransmission is completed.	

4.2 Self-Timed Transmission Command Group

This section details the Self-Timed DCPC related commands as summarized in Table 26.

Table 26: Self-Timed Transmission Command Summary

CMD Code	Short Description	Long Description	R/O
0x20	Timed Chan/BPS	Self-Timed Transmission Channel and BPS Rate (300 or 1200)	R
0x21	Timed Interval	Self-Timed Transmission Interval hh:mm:ss (00:05:00 to 24:00:00)	R
0x22	First Timed Tx	First Time of Transmission hh:mm:ss (00:00:00 to 23:59:59)	R
0x23	Timed Window	Self-Timed Transmission Window in Seconds*2 (1 to 110)	R
0x24	Timed Align	Self-Timed Transmission Window Alignment Mode (Center or Top)	R
0x25	Timed Format	Self-Timed Transmission Message Format (ASCII, PB, or Binary)	R
0x26	Timed All	All of the Self-Timed Parameters	O
0x27	Future		?
thru	Future	**NOTE: Some of these could be transmitter/mfgr specific.**	?
0x2F	Future		?

4.2.1 Timed Channel and BPS Rate Command – Required

The Timed Channel and BPS Rate command sets the GOES DCS CS2 channel and data rate for Self-Timed transmissions.

Table 27: Command Specifics – Timed Channel and BPS Rate Command

	Type/Value	Details/Notes
Command Code:	0x20	
Command Data:	2-Byte ULE	Self-Timed CS2 Channel (0, 1-266, 301-566)
	Byte ENUM	Self-Timed Data Rate 0x00=N/A, 0x01=300, 0x02=1200, 0x03-0xFF Future
Request Data:	2-Byte ULE	Current Self-Timed Channel.
	Byte ENUM	Current Self-Timed Data Rate.
Special ACK:	0x0A	Channel not supported (possible future use).
	0x0B	Data Rate not supported (possible future use).
Special Notes:	A Channel value of 0 disables Self-Timed transmissions and should be accompanied by a Data Rate setting of 0x00. Send 0x02 if the channel setting is not valid for 1200 bps.	

4.2.2 Timed Interval Command – Required

The Timed Interval command sets the transmit interval in hours, minutes and seconds (hh:mm:ss) for Self-Timed transmissions.

Table 28: Command Specifics – Timed Interval Command

	Type/Value	Details/Notes
Command Code:	0x21	
Command Data:	3-Byte Time	Self-Timed Interval
Request Data:	3-Byte Time	Current Self-Timed Interval.
Special ACK:	0x0C	Interval not supported.
Special Notes:	Byte 1 Hours, 0-24; Byte 2 Minutes, 0-59; Byte 3 Seconds, 0-59.	

4.2.3 First Timed Transmission Command – Required

The First Timed Transmission command sets the time of the first Self Timed transmission for the day within the Timed Transmit interval in hours, minutes and seconds (hh:mm:ss). The First Timed Transmission must be less than the Timed Interval.

Table 29: Command Specifics – First Timed Transmission Command

	Type/Value	Details/Notes
Command Code:	0x22	
Command Data:	3-Byte Time	First Timed Transmission
Request Data:	3-Byte Time	Current First Timed Transmission.
Special ACK 1:	0x0D	First Timed not supported by transmitter.
Special ACK 2:	0x0E	First Timed equals or exceeds Timed Interval
Special Notes:	Byte 1 Hours, 0-23; Byte 2 Minutes, 0-59; Byte 3 Seconds, 0-59.	

4.2.4 Timed Window Command – Required

The Timed Window command sets the transmit windows in seconds with a 0.5 second resolution for Self-Timed transmissions. To achieve the half second resolution, the value is in seconds multiplied by 2.

Table 30: Command Specifics – Timed Window Command

	Type/Value	Details/Notes
Command Code:	0x23	
Command Data:	Byte UINT	Timed Window *2 (1-110 seconds = 2-220)
Request Data:	Byte UINT	Current Timed Window *2
Special ACK:	0x0F	Timed Window not supported by transmitter.
Special Notes:	None	

4.2.5 Timed Align Command – Required

The Timed Align command selects where in the Timed Window the Self-Timed transmissions will occur; either in the Center or beginning at the Top of the window.

Table 31: Command Specifics – Timed Align Command

	Type/Value	Details/Notes
Command Code:	0x24	
Command Data:	Byte Flag	Timed Alignment (0x00=Top; 0xFF=Center)
Request Data:	Byte Flag	Current Align setting
Special ACK:	0x10	Alignment not supported by transmitter.
Special Notes:	None	

4.2.6 Timed Format Command – Required

The Timed Format command determines the message format for Timed Transmissions.

Table 32: Command Specifics – Timed Format Command

	Type/Value	Details/Notes
Command Code:	0x25	
Command Data:	Byte ENUM	See Table 19
Request Data:	Byte ENUM	Current Self-Timed Format
Special ACK:	0x11	Timed Format not supported by transmitter.
Special Notes:	None	

Table 33: Command Specifics – Message Format Codes

Code	Name	Description
0x00 … 0x07	Reserved	Possible Future Use
0x08	Standard ASCII	Standard or Legacy ASCII
0x09 …0x0F	Reserved	Possible Future Use
0x10	Open Binary	Standard Binary (No Compaction)
0x11	Compact PB	Binary – Pseudo Binary Compaction
0x12	Compact NA	Binary –Numeric ASCII Compaction
0x13	Compact SA	Binary – SHEF ASCII Compaction
0x14	Compact FA	Binary – Full ASCII Compaction
0x15 … 0x17	Reserved	Possible Future Use
0x18	Standard PB	Standard or Legacy Pseudo Binary
0x19 … 0xFF	Reserved	Possible Future Use

4.2.7 Timed All Command – Optional

The Timed All command can be used to set or retrieve all parameters associated with Timed transmissions.

Table 34: Command Specifics – Timed All Command

	Type/Value	Details/Notes
Command Code:	0x26	
Command Data:	2-Byte ULE	Self-Timed Channel.
	Byte ENUM	Self-Timed Data Rate.
	3-Byte Time	Self-Timed Interval
	3-Byte Time	First Self-Timed Transmission
	Byte UINT	Self-Timed Window *2
	Byte Flag	Self-Timed Alignment (0x00=Top; 0xFF=Center)
	Byte ENUM	Self-Timed Format
Request Data:	Same	Order and data format as Command Data.
Special ACKs:	0x0A-0x11	See section associated with parameter for ACK info.
Special Notes:	See section associated with parameter for data specifics.	

4.3 Random Transmission Commands

This section details the Random DCPC related commands as summarized in Table 35.

Table 35: Random Transmission Command Summary

CMD Code	Short Description	Long Description	R/O
0x30	Random Ch/BPS	Random Transmission Channel and BPS (300 or 1200)	R
0x31	Random Interval	Random Transmission Interval hh:mm:ss (00:02:30 to 24:00:00)	R
0x32	Random Percent	Randomization Percentage (10-50%)	R
0x33	Random Count	Random Transmission Count (1 to 99)	R
0x34	Random Format	Random Message Format (ASCII, PB, or Binary)	R
0x35	Random All	All of the Random Parameters	O
0x36	Future		
thru	Future	**NOTE: Some of these could be transmitter/mfgr specific.**	
0x3A	Future		
0x3B	DCPC Channel(s)	DCPC Acknowledge Tx Channel(s)	R
0x3C	DCPC Interval	DCPC Acknowledge Interval mm:ss (02:30 to 30:00)	R
0x3D	DCPC Percent	DCPC Acknowledge Randomization Percentage (10-50%)	R
0x3E	DCPC Count	DCPC Acknowledge Transmission Count (1 to 99)	R
0x3F	DCPC All	All of the DCPC Acknowledge Parameters	R

4.3.1 Random Channel and BPS Rate Command – Required

The Random Channel and BPS Rate command sets the GOES DCS CS2 channel and data rate for Random transmissions.

Table 36: Command Specifics – Random Channel and BPS Rate Command

	Type/Value	Details/Notes
Command Code:	0x30	
Command Data:	2-Byte ULE	Random CS2 Channel (0, 1-266, 301-566)
	Byte ENUM	Random Data Rate 0x00=N/A, 0x01=300, 0x02=1200, 0x03-0xFF Future
Request Data:	2-Byte ULE	Current Random Channel.
	Byte ENUM	Current Random Data Rate.
Special ACK:	0x0A	Channel not supported (possible future use).
	0x0B	Data Rate not supported (possible future use).
Special Notes:	A Channel value of 0 disables Random transmissions and should be accompanied by a Data Rate setting of 0x00. Send 0x02 if the channel setting is not valid for 1200 bps.	

4.3.2 Random Interval Command – Required

The Random Interval command sets the transmit interval in hours, minutes and seconds (hh:mm:ss) for Random transmissions.

Table 37: Command Specifics – Random Interval Command

	Type/Value	Details/Notes
Command Code:	0x31	
Command Data:	3-Byte Time	Random Interval
Request Data:	3-Byte Time	Current Random Interval
Special ACK:	0x0C	Interval not supported; e.g. exceeds max for DCP.
Special Notes:	Byte 1 Hours, 0-24; Byte 2 Minutes, 0-59; Byte 3 Seconds, 0-59.	

4.3.3 Random Percent Command – Required

The Random Percent command sets the randomization percentage applied to the Random Interval for Random transmissions.

Table 38: Command Specifics – Random Percent Command

	Type/Value	Details/Notes
Command Code:	0x32	
Command Data:	Byte UINT	Randomization Percentage (10% to 50%)
Request Data:	Byte UINT	Current Randomization Percentage
Special ACK:	0x0D	Randomization value not supported by transmitter.
Special Notes:	None	

4.3.4 Random Count Command – Required

The Random Count command sets the number of Random transmissions in a triggered Random Reporting sequence.

Table 39: Command Specifics – Random Count Command

	Type/Value	Details/Notes
Command Code:	0x33	
Command Data:	Byte UINT	Random Transmission Count (1 to 99)
Request Data:	Byte UINT	Current Random Transmission Count
Special ACK:	0x0E	Count value not supported by transmitter.
Special Notes:	None	

4.3.5 Random Format Command – Required

The Random Format command determines the message format for Random Transmissions.

Table 40: Command Specifics – Random Format Command

	Type/Value	Details/Notes
Command Code:	0x34	
Command Data:	Byte ENUM	See Table 19
Request Data:	Byte ENUM	Current Random Format
Special ACK:	0x0F	Random Format not supported by transmitter.
Special Notes:	None	

4.3.6 Random All Command – Optional

The Random All command can be used to set or retrieve all parameters associated with Random transmissions.

Table 41: Command Specifics – Random All Command

	Type/Value	Details/Notes
Command Code:	0x26	
Command Data:	2-Byte ULE	Random Channel.
	Byte ENUM	Random Data Rate.
	3-Byte Time	Random Interval
	Byte UINT	Randomization Percentage
	Byte UINT	Random Transmission Count
	Byte ENUM	Random Format
Request Data:	Same	Order and data format as Command Data.
Special ACKs:	0x0A-0x0F	See section associated with parameter for ACK info.
Special Notes:	See section associated with parameter for data specifics.	

4.3.7 DCPC Channel(s) Command – Required

The DCPC Channel(s) command sets the GOES DCS CS2 channel for Random ACK transmissions. As indicated in Table 42, the first channel must always be specified, but the second and third can be undefined. A channel value of 0 means the channel is not specified. If the second channel setting is 0, the third must also be 0.

Table 42: Command Specifics – DCPC Channel(s) Command

	Type/Value	Details/Notes
Command Code:	0x3B	
Command Data:	2-Byte ULE	First Random ACK Channel (1-266, 301-566)
	2-Byte ULE	Second Random ACK Channel (0, 1-266, 301-566)
	2-Byte ULE	Third Random ACK Channel (0, 1-266, 301-566)
Request Data:	2-Byte ULE	Current First Random ACK Channel.
	2-Byte ULE	Current Second Random ACK Channel.
	2-Byte ULE	Current Third Random ACK Channel.
Special ACK:	0x0A	First Channel not supported (possible future use).
	0x0B	Second Channel not supported (possible future use).
	0x0C	Third Channel not supported (possible future use).
	0x0D	Third Channel is not 0, when the Second is 0.
Special Notes:	A Channel value of 0 indicates the Second or Third Random ACK Channel is undefined; if the Second is 0, the Third must also be 0.	

4.3.8 DCPC Interval Command – Required

The DCPC Interval command sets the Random ACK transmit interval in minutes and seconds (mm:ss).

Table 43: Command Specifics – DCPC Interval Command

	Type/Value	Details/Notes
Command Code:	0x3C	
Command Data:	2-Byte Time	Random ACK Interval (01:00 – 15:00)
Request Data:	2-Byte Time	Current Random ACK Interval
Special ACK:	None	
Special Notes:	Byte 1 Minutes, 1-15; Byte 2 Seconds, 0-59.	

4.3.9 DCPC Percent Command – Required

The DCPC Percent command sets the randomization percentage applied to the Random ACK Interval for Random ACK transmissions.

Table 44: Command Specifics – DCPC Percent Command

	Type/Value	Details/Notes
Command Code:	0x3D	
Command Data:	Byte UINT	Randomization Percentage (10% to 50%)
Request Data:	Byte UINT	Current Randomization Percentage
Special ACK:	None	
Special Notes:	None	

4.3.10 DCPC Count Command – Required

The DCPC Count command sets the number of Random ACK transmissions in a triggered DCPC Acknowledgment sequence.

Table 45: Command Specifics – DCPC Count Command

	Type/Value	Details/Notes
Command Code:	0x3E	
Command Data:	Byte UINT	Random ACK Transmission Count (1 to 9)
Request Data:	Byte UINT	Current Random ACK Transmission Count
Special ACK:	None	
Special Notes:	None	

4.3.11 DCPC All Command – Required

The DCPC All command can be used to set or retrieve all parameters associated with Random ACK transmissions used in an DCPC Acknowledgment sequence.

Table 46: Command Specifics – Random All Command

	Type/Value	Details/Notes
Command Code:	0x3E	
Command Data:	2-Byte ULE	First Random ACK Channel (1-266, 301-566)
	2-Byte ULE	Second Random ACK Channel (0, 1-266, 301-566)
	2-Byte ULE	Third Random ACK Channel (0, 1-266, 301-566)
	2-Byte Time	Random ACK Interval (01:00 – 15:00)
	Byte UINT	Randomization Percentage (10% to 50%)
	Byte UINT	Random ACK Transmission Count (1 to 9)
Request Data:	Same	Order and data format as Command Data.
Special ACKs:	0x0A-0x0D	See section associated with parameter for ACK info.
Special Notes:	See section associated with parameter for data specifics.	

4.4 Sensor Reading and Configuration Command Group

This section provides suggested Sensor Reading and Configuration Commands. Sensor configuration is particularly unique to datalogger portion of the platform, and therefore quite possibly manufacturer specific. As such, it may be necessary that these commands have a manufacturer specific implementation. The subsections below are intended to provide examples of what could be possible and to provide a framework for further discussion.

All of the commands in this group are optional since the DCPC receiver may not be able to communicate with the datalogger portion of the platform. While a single device that combines the DCPC receiver, DCS transmitter, and the datalogger may ultimately become a reality, given the number of already deployed GOES DCS DCPs, it is expected that the initial use of the DCPC system will be accomplished by adding a separate DCPC receiver to an existing platform. Further, while many DCPs utilize a single unit that performs both the GOES DCS transmit and datalogging functions, many other platforms consist of two distinct units. As such, it may not be practical or possible for the DCPC receiver to communicate with the separate datalogger unit in and existing system. Whereas, a DCPC receiver must be able to communicate with the DCS transmitter to able to fulfill its intended purpose.

As summarized in Table 47, there are six Sensor Reading and Configuration Commands initially envisioned. The remaining ten of the Command Codes in this group could define additional common functionality similar to the first six, or could simply be reserved and assigned for datalogger and/or manufacturer specific use.

Table 47: Sensor Reading and Configuration Command Summary

CMD Code	Short Description	Long Description	R/O
0x50	Sensor Sample	Set/Request Sensor Sampling Interval Parameters	O
0x51	SDI-12 Sensor	Set/Request SDI-12 Sensor Parameters	O
0x52	Sensor String	Set/Request Sensor Name/Label/SHEF Code String	O
0x53	Timed Sensor	Set/Request Timed Buffer Sensor Parameters	O
0x54	Random Sensor	Set/Request Random Buffer Sensor Parameters	O
0x55	Random Trigger	Set/Request Random Trigger Sensor Parameters	O
0x56	Future		O
thru	Future	**NOTE: Some of these could be logger/manufacturer specific.**	O
0x5F	Future		O

4.4.1 Set/Request Sensor Sampling Interval – Optional

Table 48 shows the suggested format for the Sensor Sampling Interval command.

Table 48: Set/Request Sensor Sampling Interval

	Type/Value	Details/Notes
Command Code:	0x50	
Command Data:	Byte UINT	Sensor Number
	3-Byte Time	Sampling Interval (hh:mm:ss)
	3-Byte Time	Sampling Offset into Interval (hh:mm:ss)
	Byte UINT	Log Rate (Number of Samples between Logging)
	Byte ENUM	Sensor Type (MFGR Specific)
Request Data:	Same	Order and data format as Command Data.
Special ACK 1:	0x0A	Sensor Number Exceeds Limit
Special ACK 2:	0x0B	Sampling Interval Invalid or Not Supported
Special ACK 3:	0x0C	Sampling Offset Invalid or Not Supported
Special ACK 4:	0x0D	Log Rate Invalid or Not Supported
Special ACK 5:	0x0E	Sensor Type Invalid or Not Supported
Special Notes:		

4.4.2 Set/Request SDI-12 Sensor Parameters – Optional

Table 49 shows the suggested format for the SDI-12 Sensor Parameters command.

Table 49: Set/Request SDI-12 Sensor Parameters

	Type/Value	Details/Notes
Command Code:	0x51	
Command Data:	Byte UINT	Sensor Number
	Byte Char	SDI-12 Device Address
	Byte UINT	Measurement Number and Type (see Note 1)
	Byte UINT	Parameter Number
Request Data:	Same	Order and data format as Command Data.
Special ACK 1:	0x0A	Sensor Number Exceeds Limit
Special ACK 2:	0x0B	SDI-12 Device Address Invalid or Not Supported
Special ACK 3:	0x0C	Measurement Number Invalid or Not Supported
Special ACK 4:	0x0D	Parameter Number Invalid or Not Supported
Special Note 1:	0-9 => M; 10-19 => C i.e. normal SDI-12 Measure versus SDI-12 Concurrent Measurement	

4.4.3 Set/Request Sensor Name/Label/SHEF Code – Optional

Table 50 shows the suggested format for the Sensor Name/Label/SHEF Code command.

Table 50: Set/Request Sensor Name/Label/SHEF Code

	Type/Value	Details/Notes
Command Code:	0x52	
Command Data:	Byte UINT	Sensor Number
	String	Name/Label/SHEF Code String
Request Data:	Same	Order and data format as Command Data.
Special ACK 1:	0x0A	Sensor Number Exceeds Limit
Special ACK 2:	0x0B	String is Invalid or Not Supported
Special Notes:		

4.4.4 Set/Request Timed Buffer Sensor Parameters – Optional

Table 51 shows the suggested format for the Timed Buffer Sensor Parameters command that defines how sensor data is configured in the Timed Data Buffer.

Table 51: Set/Request Sensor Timed Buffer Sensor Parameters

	Type/Value	Details/Notes
Command Code:	0x53	
Command Data:	Byte UINT	Timed Parameter Number (Position in Buffer)
	Byte UINT	Sensor Number
	Byte UINT	Sensor Reading Count (Number of readings in Buffer)
	Byte BM	Sensor Reading Flags (Mfgr Specific)
	Byte UINT	Field Width (Number chars/bits for readings in Buffer)
	4-Byte Float	Reading Scale (for Pseudo Binary and Binary)
	4-Byte Float	Reading Offset (for Pseudo Binary and Binary)
Request Data:	Same	Order and data format as Command Data.
Special ACK 1:	0x0A	Parameter Number Exceeds Limit
Special ACK 2:	0x0B	Sensor Number Invalid or Not Defined
Special ACK 3:	0x0B	Sensor Reading Count Invalid or Not Supported
Special ACK 4:	0x0C	Sensor Reading Flags Invalid or Not Supported
Special ACK 5:	0x0D	Field Width Invalid or Not Supported
Special ACK 6:	0x0E	Scale Invalid or Not Supported
Special ACK 7:	0x0F	Offset Invalid or Not Supported
Special Notes:	For ASCII and Pseudo Binary, the Field Width is in characters. For Binary, the Field Width is in bits.	

4.4.5 Set/Request Random Buffer Sensor Parameters – Optional

Table 52 shows the suggested format for the Random Buffer Sensor Parameters command that defines how sensor data is configured in the Random Data Buffer.

Table 52: Set/Request Sensor Random Buffer Sensor Parameters

	Type/Value	Details/Notes
Command Code:	0x54	
Command Data:	Byte UINT	Random Parameter Number (Position in Buffer)
	Byte UINT	Sensor Number
	Byte UINT	Sensor Reading Count (Number of readings in Buffer)
	Byte BM	Sensor Reading Flags (Mfgr Specific)
	Byte UINT	Field Width (Number chars/bits for readings in Buffer)
	4-Byte Float	Reading Scale (for Pseudo Binary and Binary)
	4-Byte Float	Reading Offset (for Pseudo Binary and Binary)
Request Data:	Same	Order and data format as Command Data.
Special ACK 1:	0x0A	Parameter Number Exceeds Limit
Special ACK 2:	0x0B	Sensor Number Invalid or Not Defined
Special ACK 3:	0x0B	Sensor Reading Count Invalid or Not Supported
Special ACK 4:	0x0C	Sensor Reading Flags Invalid or Not Supported
Special ACK 5:	0x0D	Field Width Invalid or Not Supported
Special ACK 6:	0x0E	Scale Invalid or Not Supported
Special ACK 7:	0x0F	Offset Invalid or Not Supported
Special Notes:	For ASCII and Pseudo Binary, the Field Width is in characters. For Binary, the Field Width is in bits.	

4.4.6 Set/Request Random Trigger Sensor Parameters – Optional

Table 53 shows the suggested format for the Random Trigger Sensor Parameters command that defines how sensor data can/should trigger a Random Report Sequence.

Table 53: Set/Request Sensor Random Trigger Sensor Parameters

	Type/Value	Details/Notes
Command Code:	0x55	
Command Data:	Byte UINT	Random Trigger Number Identifier
	Byte UINT	Sensor Number
	Byte ENUM	Trigger Type (Mfgr Specific; e.g. high, low, delta)
	4-Byte Float	Trigger Limit
Request Data:	Same	Order and data format as Command Data.
Special ACK 1:	0x0A	Trigger Number Exceeds Limit
Special ACK 2:	0x0B	Sensor Number Invalid or Not Defined
Special ACK 3:	0x0B	Trigger Type Count Invalid or Not Supported
Special ACK 4:	0x0E	Trigger Limit Invalid or Not Supported
Special Notes:		

4.5 Special Command Group

The commands detailed in this section are considered “Special Use Case” commands and are summarized in Table 54.

Table 54: Special Command Summary

CMD Code	Short Description	Long Description	R/O
0xF0	Firmware Patch T	Patch Firmware to Transmitter	O
0xF1	Firmware Patch R	Patch Firmware to DCPC Receiver	O
0xF2	Firmware Patch D	Patch Firmware to Datalogger	O
0xF3	Direct Command T	Send direct Transmitter to Logger – In native Transmitter format.	O
0xF4	Direct Command R	Special command for DCPC Receiver – In manufacturer format.	O
0xF5	Direct Command D	Send direct command to Datalogger – In native Datalogger format.	O
0xF6	Future		?
thru	Future	**NOTE: Some of these could be system/manufacturer specific.**	?
0xFE	Future		?
0xFF	Extended Cmd	This is reserved for future use for extended commands	R

4.5.1 Firmware Patch X Commands – Optional

This section details the three optional Firmware Patch commands listed in Table 54, each one targeted a different primary element of a DCPC capable GOES Data Collection Platform (DCP). Specifically, the three primary elements are the GOES DCS Transmitter, the GOES DCPC Receiver, and the DCP’s Datalogger.

While it is conceivable that in the future that a DCPC capable DCP could be a highly integrated unit that combines all three of the primary elements into a single product or design, there are two reasons why it makes sense to have separate commands for the separate primary elements.

First, in the nearer term, it is expected that the DCPC Receiver will be a separate addon unit. Also, while many existing DCPs are a single unit consisting of a combined Datalogger and GOES DCS Transmitter, there are also many existing and deployed DCPs that consist of a separate Datalogger and GOES DCS Transmitter.

Second, even if a fully integrated unit consisting of all three elements is ultimately realized, it is quite possible that separate microcontrollers with separate firmware images could be utilized internally to perform these functions.

As such, having commands that specifically target these primary elements is a reasonable approach; both for now and in the future.

Due to the slow data rate, it is not currently envisioned that a complete update to anyone of these three primary elements will be logistically possible due to the typical size of embedded firmware in today’s world. Firmware images are quite often well over 100 kilobytes or more, and even with the Multiple Packet Command format only a maximum of 15,872 data bytes can be sent.

However, being able to patch the firmware can be possible since many firmware images can be partially erased and partially rewritten. Such a mechanism would allow a critical

firmware bug to be addressed by only changing a small portion of the firmware; i.e. patching.

Since the mechanism to make a firmware patch will be highly device and manufacturer dependent, it is not possible to specifically define the Command Data or Result Data format for these commands. However, Table 55 shows the basic DCPC command structure for these optional commands.

Table 55: Command Specifics – Firmware Patch Command

	Type/Value	Details/Notes
Command Code:	0xFX	X=0 => T; X=1 => R; X=2 => D
Command Data:	Many Bytes	Format Device/Manufacturer Specific
Result Data:	0-68 Bytes	Device/Manufacturer Specific
Special ACK:	0x0A	Firmware Patch Failed
Special Notes:	Will typically require use of Multiple Packet Command	

4.5.2 Direct Command X Commands – Optional

As with the Firmware Patch commands detailed in the previous section, the Direct Commands are individually targeted at a specific primary element of a DCPC capable DCP; i.e. the GOES DCS Transmitter, the GOES DCPC Receiver, and the DCP's Datalogger. The reasoning for having three separate commands is similar to the logic presented in the previous section.

The Direct Command allows the user, via the DCPC system, to send one or more commands to one of the three primary elements in the native command format of the targeted unit. In others words, the Command Data for these commands is manufacturer and device specific; as is the specific use and intention of the actual native command(s).

As such, like the Firmware Patch commands, it is not possible to specifically define the data format for these commands. However, Table 56 shows the basic DCPC command structure for these optional commands.

It is left up to each DCPC receiver manufacturer to determine which, if any, of these commands to implement, and what the Command Data and Result Data format will be. Implementing one of these commands does not require that all three to be implemented; nor is a requirement that Command Data and Result Data format be common between the commands if multiple commands are implemented.

Table 56: Command Specifics – Direct Command

	Type/Value	Details/Notes
Command Code:	0xFX	X=3 => T; X=4 => R; X=5 => D
Command Data:	Many Bytes	Format Device and Manufacturer Specific
Result Data:	0-68 Bytes	Device and Manufacturer Specific
Special ACK:	0x0A	Command(s) issued, but negative response received.
Special Notes:	Will typically require use of Multiple Packet Command	

4.5.3 Extended Command – Required

The Extended Command Code does not define a specific command, but instead simply provides a mechanism for future expansion of the DCPC Command. Given how many command codes that are yet to be defined, use of Extended Commands is not envisioned to happen in the near future, if at all, but is still specified at this time to allow its future use should the need arise.

It is also envisioned that the Extended Command mechanism could be utilized for manufacturer specific functionality.

The basic concept of the Extended Command functionality is that if this Command Code is received the DCPC receiver must look into the first byte(s) of the Command Data for the actual command designation.

Until such time as the Extended Command is defined, DCPC receiver shall simply respond with the ACK Code ox 0x01, i.e. Command Unknown, to ensure this functionality can be added in the future.

5 Complete Command Summary Table

Table 57: Complete DCPC Command Summary

CMD Code	Short Description	Long Description	R/O
Control/Status Commands			
0x00	Fill	Data has no meaning; it is used as a fill between commands	R
0x01	Ping	Requests a response from the platform to check if it is still active.	R
0x02	Software Reset	DCP must execute software reset after acknowledgement.	R
0x03	Hardware Reset	DCP should execute a hard reset after acknowledgement.	O
0x04	Disable Timed	Disable Self-Timed transmissions until specified date/time	R
0x05	Enable Timed	Enable Self-Timed transmissions (use after indefinite disable).	R
0x06	Disable Random	Disable Random transmissions until specified date/time	R
0x07	Enable Random	Enable Random transmissions (use after indefinite disable).	R
0x08	Enb/Dis DCP	Enable/Disable the DCP (if supported).	O
0x09	Failsafe Reset	Reset transmitter failsafe.	R
0x0A	Transmitter Status	Send DCP transmitter status and key performance metrics.	R
0x0B	Receiver Status	Send DCPC receiver status and key performance metrics.	R
0x0C	Set Platform ID	Set 32-Bit DCP Address	R
0x0D	Receiver Listen	Set DCPC receiver listen (aka power up) mode/times.	R
0x0E	Force GPS Sync	Force a GPS Sync and report result.	R
0x0F	Lat/Lon/TxID	Initiate a Lat/Lon/TxID Report Sequence	O
0x10	Resend Timed Tx	Resend a Self-Timed Message on Specified Channel	O
0x11	Future		?
thru	Future	**NOTE: Some of these could be transmitter specific.**	?
0x1F	Future		?
Self-Timed Transmission Commands			
0x20	Timed Chan/BPS	Self-Timed Transmission Channel and BPS Rate (300 or 1200)	R
0x21	Timed Interval	Self-Timed Transmission Interval hh:mm:ss (00:05:00 to 24:00:00)	R
0x22	First Timed Tx	First Time of Transmission hh:mm:ss (00:00:00 to 23:59:59)	R
0x23	Timed Window	Self-Timed Transmission Window in Seconds*2 (1 to 110)	R
0x24	Timed Align	Self-Timed Transmission Window Alignment Mode (Center or Top)	R
0x25	Timed Format	Self-Timed Transmission Message Format (ASCII, PB, or Binary)	R
0x26	Timed All	All of the Self-Timed Parameters	O
0x27	Future		?
thru	Future	**NOTE: Some of these could be transmitter specific.**	?
0x2F	Future		?
Random Transmission Command			
0x30	Random Ch/BPS	Random Transmission Channel and BPS (300 or 1200)	R
0x31	Random Interval	Random Transmission Interval hh:mm:ss (00:02:30 to 24:00:00)	R
0x32	Random Percent	Randomization Percentage (10-50%)	R
0x33	Random Count	Random Transmission Count (1 to 99)	R
0x34	Random Format	Random Message Format (ASCII, PB, or Binary)	R
0x35	Random All	All of the Random Parameters	O
0x36	Future		
thru	Future	**NOTE: Some of these could be transmitter specific.**	
0x3A	Future		
0x3B	DCPC Channel(s)	DCPC Acknowledge Tx Channel(s)	R
0x3C	DCPC Interval	DCPC Acknowledge Interval mm:ss (02:30 to 30:00)	R
0x3D	DCPC Percent	DCPC Acknowledge Randomization Percentage (10-50%)	R
0x3E	DCPC Count	DCPC Acknowledge Transmission Count (1 to 99)	R
0x3F	DCPC All	All of the DCPC Acknowledge Parameters	R

CMD Code	Short Description	Long Description	R/O
Possible Future Command Codes			
0x40	Future		?
thru	Future	**NOTE: For Future Use**	?
0x4F	Future		?
Sensor Reading and Configuration Commands			
0x50	Sensor Sample	Set/Request Sensor Sampling Interval Parameters	O
0x51	SDI-12 Sensor	Set/Request SDI-12 Sensor Parameters	O
0x52	Sensor String	Set/Request Sensor Name/Label/SHEF Code String	O
0x53	Timed Sensor	Set/Request Timed Buffer Sensor Parameters	O
0x54	Random Sensor	Set/Request Random Buffer Sensor Parameters	O
0x55	Random Trigger	Set/Request Random Trigger Sensor Parameters	O
0x56	Future		O
thru	Future	**NOTE: Some of these could be logger/manufacturer specific.**	O
0x5F	Future		O
Possible Future Command Codes			
0x60	Future		?
thru	Future	**NOTE: For Future Use**	?
0xEF	Future		?
Special Commands			
0xF0	Firmware Patch T	Patch Firmware to Transmitter	O
0xF1	Firmware Patch R	Patch Firmware to DCPC Receiver	O
0xF2	Firmware Patch D	Patch Firmware to Datalogger	O
0xF3	Direct Command T	Send direct Transmitter to Logger – In native Transmitter format.	O
0xF4	Direct Command R	Special command for DCPC Receiver – In manufacturer format.	O
0xF5	Direct Command D	Send direct command to Datalogger – In native Datalogger format.	O
0xF6	Future		?
thru	Future		?
0xFE	Future		?
0xFF	Extended Cmd	This is reserved for future use for extended commands	R

Table 57 above provides a complete DCPC Command summary for reference. As can be readily seen the vast majority of the command codes have yet to be defined, and are reserved for future use.