|  |
| --- |
| 3D Printer Module Message Document |
|  |
| 3D Printer  Module Message Specification |

|  |  |
| --- | --- |
| Document Number: |  |
| Version: | 1.0 |
| Publication Date: | 06/28/2025 |

**Table Of Contents**

[1 Introduction 5](#_Toc202436038)

[**1.1** Revision History 5](#_Toc202436039)

[**1.2** References 5](#_Toc202436040)

[**1.3** Glossary/Acronyms 5](#_Toc202436041)

[**1.4** Document Conventions 5](#_Toc202436042)

[2 Features Overview and Host Usage 6](#_Toc202436043)

[**2.1** 6](#_Toc202436044)

[**2.2** 6](#_Toc202436045)

[**2.3** 6](#_Toc202436046)

[3 Message Processing 7](#_Toc202436047)

[**3.1** Error Responses 7](#_Toc202436048)

[**3.2** Message Extensions 7](#_Toc202436049)

[4 Common Message Fields 8](#_Toc202436050)

[**4.1** Sync 8](#_Toc202436051)

[**4.2** NumBytes 8](#_Toc202436052)

[**4.3** TransactionID 8](#_Toc202436053)

[**4.4** CmdType 8](#_Toc202436054)

[**4.5** CmdID 9](#_Toc202436055)

[**4.6** Checksum 9](#_Toc202436056)

[**4.7** Parameter field 9](#_Toc202436057)

[**4.8** Message Format 9](#_Toc202436058)

[5 Module Messages 10](#_Toc202436059)

[**5.1** Module Identify Command 10](#_Toc202436060)

[**5.2** Module Identify Response 10](#_Toc202436061)

[**5.3** Generic Response 11](#_Toc202436062)

[**5.4** Power Down Dispatch 12](#_Toc202436063)

[**5.5** Power Down Response 12](#_Toc202436064)

[**5.6** Package Dispatch 13](#_Toc202436065)

[**5.7** Package Response 14](#_Toc202436066)

[**5.8** Package Error Response 16](#_Toc202436067)

[**5.9** IP Device Authentication Dispatch 17](#_Toc202436068)

[**5.10** IP Device Authentication Response 18](#_Toc202436069)

[**5.11** IP Device Authentication Error Response 18](#_Toc202436070)

[**5.12** Authorization Pellet Dispatch 19](#_Toc202436071)

[**5.13** Authorization Pellet Response 20](#_Toc202436072)

[**5.14** Authorization Pellet Error Response 20](#_Toc202436073)

[**5.15** CA Container Dispatch 21](#_Toc202436074)

[**5.16** CA Container Error Response 22](#_Toc202436075)

[**5.17** Tunnel Dispatch 23](#_Toc202436076)

[**5.18** Tunnel Response 23](#_Toc202436077)

[6 Status Messages 25](#_Toc202436078)

[**6.1** Time Dispatch 25](#_Toc202436079)

[**6.2** System Status Dispatch 26](#_Toc202436080)

[**6.3** Subscription Status Dispatch 27](#_Toc202436081)

[**6.4** Authorization Status Dispatch 29](#_Toc202436082)

[**6.5** Channel Status Dispatch 30](#_Toc202436083)

[**6.6** Event Report Dispatch 31](#_Toc202436084)

[**6.7** Status Monitor Dispatch 32](#_Toc202436085)

[**6.8** Status Monitor Error Response 33](#_Toc202436086)

[**6.9** Antenna Aiming Dispatch 34](#_Toc202436087)

[**6.10** Signal Status Summary Dispatch 35](#_Toc202436088)

[7 Audio Services Management Messages 41](#_Toc202436089)

[**7.1** Audio Select Dispatch 41](#_Toc202436090)

[**7.2** Audio Select Error Response 43](#_Toc202436091)

[**7.3** Now Playing Dispatch 44](#_Toc202436092)

[**7.4** Playback Control Dispatch 47](#_Toc202436093)

[**7.5** Playback Control Error Response 48](#_Toc202436094)

[**7.6** Smart Favorites Configure Dispatch 49](#_Toc202436095)

[**7.7** Smart Favorites Configure Error Response 50](#_Toc202436096)

[**7.8** Sequential Content Dispatch 51](#_Toc202436097)

[**7.9** SportsFlash Configure Dispatch 53](#_Toc202436098)

[**7.10** SportsFlash Configure Error Response 54](#_Toc202436099)

[**7.11** SportsFlash Event Dispatch 54](#_Toc202436100)

[**7.12** TuneMix Content Dispatch 55](#_Toc202436101)

[**7.13** Alert Packet Dispatch 56](#_Toc202436102)

[**7.14** Alert Packet Error Response 58](#_Toc202436103)

[**7.15** Ad Configure Dispatch 59](#_Toc202436104)

[**7.16** Ad Configure Error Response 60](#_Toc202436105)

[**7.17** Ad Control Dispatch 61](#_Toc202436106)

[**7.18** Ad Control Error Response 62](#_Toc202436107)

[**7.19** Metadata Reference Dispatch 62](#_Toc202436108)

[**7.20** Lookahead Metadata Reference Dispatch 64](#_Toc202436109)

[8 Firmware Update Messages 65](#_Toc202436110)

[**8.1** FWU Start Dispatch 65](#_Toc202436111)

[**8.2** FWU Start Response 66](#_Toc202436112)

[**8.3** FWU Start Error Response 66](#_Toc202436113)

[**8.4** FWU Packet Dispatch 67](#_Toc202436114)

[**8.5** FWU Packet Response 68](#_Toc202436115)

[**8.6** FWU Packet Error Response 68](#_Toc202436116)

[**8.7** FWU Status Dispatch 69](#_Toc202436117)

[9 Production Support Messages 71](#_Toc202436118)

[**9.1** Set Production Data Dispatch 71](#_Toc202436119)

[**9.2** Set Production Data Error Response 71](#_Toc202436120)

[**9.3** Set Service Data Dispatch 72](#_Toc202436121)

[**9.4** Set Service Data Error Response 73](#_Toc202436122)

[**9.5** Get Service Data Dispatch 74](#_Toc202436123)

[**9.6** Get Service Data Response 74](#_Toc202436124)

[**9.7** Tuner Calibration Dispatch 75](#_Toc202436125)

[**9.8** ADI Tuner Calibration Response 75](#_Toc202436126)

[**9.9** Broadcom Tuner Calibration Response 76](#_Toc202436127)

[**9.10** NXP Tuner Calibration Response 77](#_Toc202436128)

[**9.11** GPIO Control Dispatch 78](#_Toc202436129)

[**9.12** GPIO Control Response 79](#_Toc202436130)

[**9.13** Set Operating Mode Dispatch 79](#_Toc202436131)

[**9.14** Set Operating Mode Response 80](#_Toc202436132)

[**9.15** UART1 Control Dispatch 81](#_Toc202436133)

[**9.16** UART1 Control Response 82](#_Toc202436134)

[**9.17** DDR3 Test Control Dispatch 82](#_Toc202436135)

[**9.18** DDR3 Test Control Response 83](#_Toc202436136)

[**9.19** Maintenance Status Dispatch 84](#_Toc202436137)

[**9.20** Maintenance Status Response 84](#_Toc202436138)

[**9.21** C/N Status Dispatch 86](#_Toc202436139)

[**9.22** Messaging Status Dispatch 87](#_Toc202436140)

[**9.23** NVM Test Control Dispatch 88](#_Toc202436141)

[**9.24** NVM Test Control Response 89](#_Toc202436142)

[13 Engineering Messages – Memory 91](#_Toc202436143)

[13.1 Write Memory Dispatch 91](#_Toc202436144)

[13.2 Write Memory Error Response 92](#_Toc202436145)

[13.3 Read Memory Dispatch 92](#_Toc202436146)

[13.4 Read Memory Response 93](#_Toc202436147)

[13.5 Write Serial Flash Dispatch 94](#_Toc202436148)

[13.6 Write Serial Flash Error Response 95](#_Toc202436149)

[13.7 Read Serial Flash Dispatch 95](#_Toc202436150)

[13.8 Read Serial Flash Response 96](#_Toc202436151)

# Introduction

This document describes the first edition of 3D\_Printer\_GCODE\_Commander and the 3D\_Printer\_Module, which will both be used to operate a 3D printer with X, Y, Z axis stepper motors given a GCODE file with valid GCODE commands.

## Revision History

## References

## Glossary/Acronyms

|  |  |
| --- | --- |
| – (hyphen character) | Symbol to represent an inclusive range of values. |
| : (colon character) | Symbol to represent a range of ordered values with first value representing the first value in the group. For example, 0:3 represents values 0, 1, 2, and 3 in that order; 7:4 represents values 7, 6, 5, and 4 in that order. |
| 0x | Prefix indicating hexadecimal notation. |
| LSB | Least Significant Bit |
| Module | The *Module* refers to the 3D\_Printer\_module which runs on a ATSAMD21G18A processor on a Arduino M0 Pro board. The Module is controlled by the 3D\_Printer\_GCODE\_Commander, which supplies it with instructions derived from GCODE commands. |
| Host | The 3D\_Printer\_GCODE\_Commander desktop application. This application translates GCODE commands into instructions that the Module will be able to understand. |
| UART | Universal Asynchronous Receiver/Transmitter |
| User | The 3D printer operator that interacts with the Host User Interface. |

## Document Conventions

All numerical values are represented in decimal radix unless stated otherwise or prefixed with “0x” notation representing hexadecimal notation. Numeric values are 8-bit, 16-bit, or 32-bit unsigned or signed (2’s complement) integers. Bit-0 is always the least-significant bit of the least-significant byte of the numeric.

|  |  |  |
| --- | --- | --- |
| **Type (C/C#)** | **# Bytes** | **Definition** |
| uint8 | 1 | integer, 8-bit, unsigned |
| int8, byte | 1 | integer, 8-bit, signed (two’s complement) |
| uint16, ushort | 2 | integer, 16-bit, unsigned |
| int16, short | 2 | integer, 16-bit, signed (two’s complement) |
| uint32, uint(C#) | 4 | integer, 32-bit, unsigned |
| int32, int(C#) | 4 | integer, 32-bit, signed (two’s complement) |

Bitfields are a specified number of bits that are not even multiples of 8 bits. Bit 0 of a bitfield refers to the least-significant bit of the last (rightmost) byte in the representation. All bitfields are designed to fit within an integral number of 8-bit bytes.

# Features Overview and Host Usage

This section provides a brief introduction to the Module, and a brief description of how the Host is expected to use the Module’s capabilities with the Host main application.

## 

## 

## 

# Message Processing

There are two types of messages in this two device system:

1. Dispatch: Sent from the Host to the module, contains instructions for the module.
2. Response: Sent from module to the Host as a reply to a command.

Most Commands sent by the Commander Main Application generate a corresponding Acknowledgement from the receiving device(module).

## Generic Response

A Generic Response can be sent by the Host or Module in cases where there is no pre-defined response. It will contain the same incremental transaction ID and contain a status field to show if a previous command/response was processed successfully.

## Error Response

A Error Responses can be sent by the Host or Module in cases where there an error occurred in processing a Dispatch. It will contain the same incremental transaction ID and contain a parameter field with error codes.

# Common Message Fields

This section provides details of parameters common to more than one message. Sections 4.1-4.6 are parts of the message called the ‘Base Message’. All possible messages shall contain the Base Message. Any additional information must be held in the bytes proceeding the Base Message.

## Sync

The Sync provides a fixed bit pattern to identify the start of a message within the byte stream. The Sync is 0xB7 and is the same for all messages. There is nothing to prevent the Sync from otherwise being present in the byte stream, so other fields (e.g., the checksum) within the message must also be checked to confirm the message is valid.

## NumBytes

NumBytes is the number of bytes in the indicated message. It includes the Sync, NumBytes, Checksum, TransactionID, CmdType, CmdID, and all message-specific fields. In all cases, the minimum value of NumBytes is 9, also called the baseMessage size. The number of Message parameters may range from 0 to 5+, so a message’s size can vary depending on the type of command it contains.

## TransactionID

A TransactionID is an 8-bit value included in each message. The TransactionID in a Dispatch will be echoed by the receiver of the message in the corresponding Response(Acknowledgment), allowing the Sender(Host) to match up messages with their replies:

The Host shall set the TransactionID to 0 in all Module Configure Dispatches sent to the Module. For subsequent Dispatches, the Host shall increment the TransactionID by 1 and roll it over back to 0 when it reaches 256 (i.e., it takes on values of 0-255).

The Module shall create an acknowledgement response for the command and assign to it the same TransactionID as the command it received. Then it will be sent to the Host.

The TransactionID can also be used to detect when messages have been “lost”.

## CmdType

A unique 8-bit CmdType is assigned to each message to identify it, allowing interpretation of the associated message fields.

| **Value** | **CmdType** | **Function** |
| --- | --- | --- |
| 0 | BEGIN | Used to indicate to the module to begin the printing process |
| 1 | END | Terminate the printing process |
| 2 | PAUSE | Pause the printing process |
| 3 | DX | If sent to the module, it is a Diagnosis Command.  If sent to the Host, it is a Diagnosis Response. |
| 4 | PD | Power Down Command |
| 5 | ID | Dispatch contains a Identify command, Response contains a Identify response message. |
| 6 | MCF | Dispatch contains a Module config command, with parameters. Responses with this type have a status and no parameters. |
| 7 | G\_CMD | Dispatch contains a G-Type GCODE |
| 8 | M\_CMD | Dispatch contains a M-Type GCODE |
| 9 | GEN\_RSP | Response is a Generic Response |
| 10 | STS\_RSP | Response is a Status Message Response |
| 11 | DYN\_RSP | Response is a Dynamic Message which contains additional parameters |
| 12 | ERR | Used in both Dispatches and Responses to indicate a critical error. |

## CmdID

A 2 byte value to determine a specific GCODE command. Used in conjunction with the CmdType. G and M codes will have a identifier commonly appended to their type, which is to be stored in CmdID. Non-GCODE commands have designated pre-set ID’s.

## Checksum

A Checksum is included in each message to allow verification that the message has not been corrupted. An algorithm will generates a 16-bit checksum, computed over all bytes in the message with the checksum bytes set to 0.

## Parameter field

An optional byte field appended to the end of the message carrying parameters that a GCODE may need. These are stored in 4 byte float values as the parameter values can have decimals. Each 4 byte float value will be prefaced by a byte value that contains an ASCII code for a parameter type, e.g. X, Y, Z, E … etc. A message can have more than one parameter.

To determine how many parameter values are in a message, do : (numBytes – baseMessageSize) / (sizeof(float) +1).

## Message Format

Below is the format of what a message to the module will look like, in a byte stream format.

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | - |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | - |
| 7-8 | CmdID | uint16 | - |
| 8+ | Optional Command Parameters: | uint8  +  uint32 | See Section 4.7 |

# Module Messages

The Host will send Dispatch messages to the module, which the module will respond to. The module shall send two responses for each Dispatch message. The first is a Generic Response, indicating the module received the Dispatch, the second is a Module Status Response, which will contain information regarding errors that the module encountered while processing the message.

## Module Identify Dispatch

The Host sends the ‘Module Identify Dispatch’ to the Module to establish communication with the Module and to configure its operational parameters.

The Module ignores any data received on the UART interface during its startup process. Once it is fully operational, it will begin interpreting data on the UART interface. The Module Identify Command is the first message sent from the Host that the Module will accept after the Module is reset.

If the Module is not ready to process the Module Identify Command, then the Module shall not send any response to that Dispatch.

Message Usage: Establish Module Connection

Message Direction: Sent to Module from Host

Response: Module Identify Response

Response Timeout: 3 seconds

Minimum **NumBytes**: 8

Maximum **NumBytes**: 9

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | 0 |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | 0 |

## Module Identify Response

The Module sends the ‘Module Identify Response’ to the Host to establish communication with the Module and to configure its operational parameters.

The Module ignores any data received on the UART interface during its startup process. Once it is fully operational, it will begin interpreting data on the UART interface. The Module Identify Command is the first message sent from the Host that the Module will accept after the Module is reset.

If the Module is not ready to process the Module Identify Command, then the Module shall not send any response to that Dispatch.

Message Usage: Establish Module Connection

Message Direction: Sent to Module from Host

Response: Module Identify Response

Response Timeout: 1 seconds

Minimum **NumBytes**: 9

Maximum **NumBytes**: 50

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | 0 |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | 0x8000 |
| 9 | Firmware Field Identifier | uint8 |  |
| 10-13 | Firmware Version | uint32 | See below |
| 14 | Capabilities Field Identifier | uint8 |  |
| 15-18 | Capabilities of the Printer | uint32 | See below |

**Firmware Version:**

Major number of the firmware version is stored in the bits (24-16)

Minor Number is stored in the bits (15-8)

Revision number stored in the bits (7-0)

**Capabilities:**

| **Byte 3** | **Byte 2** | **Byte 1** | **Byte 0** |
| --- | --- | --- | --- |
| General | Z axis | Y axis | X axis |

X axis: 0: cannot print on X, 1: can print on X

Y axis: 0: cannot print on Y, 1: can print on Y

Z axis: 0: cannot print on Z, 1: can print on Z

General: Place values for tool settings….

## Module Config Dispatch

The Host sends this dispatch to the Module to configure the print settings. The module will initialize its global variables, motors, and tool, and move the motors to a neutral position. The module will respond with a Generic Response to indicate that the message was received, and a Module Status Response to indicate that the message was processed correctly.

Message Usage: Configure printing

Message Direction: Sent to Module from Host

Response: Module Config Response

Response Timeout: 10 seconds

Minimum **NumBytes**: 8

Maximum **NumBytes**: 9

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | Incremental |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | 1 |
| 8+ | Parameters: see below. | uint8  +  uint32 | See Section 4.7 |

**Parameters:**

These parameters should be the x, y, z axis boundaries the print will take, a neutral location where the tool can reside when not printing, and the type of tool.

TOD: current system time, to initialize module clock.

## Module Config Response

The Module sends this response to the Host to confirm the configuration. The module will report a status that should be used to determine if the Module Config Dispatch was processed correctly.

Message Usage: Configure printing

Message Direction: Sent to Host from Module

Minimum **NumBytes**: 8

Maximum **NumBytes**: 10

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | TXN ID of Command |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | 0x8001 |
| 9 | Status: see below | uint8  +  uint32 | See Section 4.7 |

**Status**:

Indicates status of the processed Dispatch. The valid values are:

0 = Dispatch received and processed

Any non-zero value or the message not being received means there was an error on the Module side.

## Module Command Dispatch

The Host sends this dispatch to the Module signal a variety of functions, dictated by CmdType values. The module will respond with a Module Command Response to indicate that the message was received and processed correctly.

Message Usage: Do a printing step

Message Direction: Sent to Module from Host

Response: Module Command Response

Response Timeout: 10 seconds

Minimum **NumBytes**: 9

Maximum **NumBytes**: 50

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | Incremental |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Below |
| 7-8 | CmdID | uint16 | 2 |

**CmdType:**

| **Value** | **CmdType** | **Function** |
| --- | --- | --- |
| 0 | BEGIN | Used to indicate to the module to begin the printing process |
| 1 | END | Terminate the printing process |
| 2 | PAUSE | Pause the printing process |
| 12 | ERR | Used in both Dispatches and Responses to indicate a critical error. |

## Module Command Response

For Commands sent to the module, only an acknowledgement that the Dispatch was received is necessary to send back to the sender. CmdType and CmdID values in the response are mirrored from the Generic Command sent to the Module. A status byte is appended to the end of the base message to indicate a failure to process the message. Should the Host receive a status that’s nonzero(fail), a diagnosis should be done on the module.

Message Usage: Respond to Module Command Dispatch

Message Direction: Sent from Module, Sent to Host

Minimum **NumBytes**: 9

Maximum **NumBytes**: 10

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | TXN ID of Command |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | 0x8002 |
| 9 | Status | uint8 | See below |

**Status**:

Indicates status of the processed Dispatch. The valid values are:

0 = Dispatch received and processed

Any non-zero value or the message not being received means there was an error on the Module side.

## Print Command Dispatch

The Host sends this dispatch to the Module to start a new printing step. This dispatch carries a set of parameters pertaining to the GCODE command.

Message Usage: Respond to Module Command Dispatch

Message Direction: Sent from Module, Sent to Host

Minimum **NumBytes**: 9

Maximum **NumBytes**: 50

Response: Print Command Response

Response Timeout: 10 seconds

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | TXN ID of Command |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See below |
| 7-8 | CmdID | uint16 | See Section 4.5 |
| 8+ | Parameters: see below. | uint8  +  uint32 | See Section 4.7 |

**CmdType:**

|  |  |  |
| --- | --- | --- |
| **Value** | **CmdType** | **Function** |
| 3 | G | Dispatch contains a G-Type GCODE |
| 4 | M | Dispatch contains a M-Type GCODE |

**Status**:

Indicates status of the processed Dispatch. The valid values are:

0 = Dispatch received and processed

Any non-zero value or the message not being received means there was an error on the Module side.

**Optional Parameters:**

GCODE command specific parameters

## Print Command Response

For G, M Commands sent to the module, only an acknowledgement that the Dispatch was received is necessary to send back to the sender. CmdType and CmdID values in the response are mirrored from the Command sent to the Module. A status byte is appended to the end of the base message to indicate a failure to process the message. Should the Host receive a status that’s nonzero(fail), a diagnosis should be done on the module.

Message Usage: Respond to Module Command Dispatch

Message Direction: Sent from Module, Sent to Host

Minimum **NumBytes**: 9

Maximum **NumBytes**: 10

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | TXN ID of Command |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | See Section 4.5 |
| 9 | Status | uint8 | See below |

**Status**:

Indicates status of the processed Dispatch. The valid values are:

0 = Dispatch received and processed

Any non-zero value or the message not being received means there was an error on the Module side.

## Diagnosis Dispatch

The Commander sends this dispatch to the Module to get any error codes the module may have encountered while processing dispatches. The module will respond with a Diagnosis Response to indicate that the dispatch was received and processed.

Message Usage: Get Errors in the module

Message Direction: Sent to Module from Host

Response: Diagnosis Response

Response Timeout: 5 seconds

Minimum **NumBytes**: 9

Maximum **NumBytes**: 9

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | 0 |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | 4 |

## Diagnosis Response

The module sends this response to the Host with any error codes the module may have encountered while processing information. This response is sent after the Diagnosis Dispatch is received and processed.

Message Usage: Get Errors in the module

Message Direction: Sent from Module to Host

Minimum **NumBytes**: 9

Maximum **NumBytes**: 45

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | 0 |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | 0x8004 |
| 8+ | Error Codes: See below. | uint8  +  uint32 | See Section 4.7 |

**Error Codes**:

Indicates status of the Motors and their coordinates, the temperature and status of the tool, and any error codes the module found while processing previous commands.

Values will be 4 byte values, prefaced by a byte identifier to indicate what type of problem is being discussed.

## Power Down Dispatch

The Host sends the Power Down Dispatch to the Module to have it perform an orderly shutdown of the Module.

Message Usage: Shut down Module System

Message Direction: Sent to Module

Response: Power Down Response

Response Timeout: 10 seconds

Minimum **NumBytes**: 8

Maximum **NumBytes**: 10

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | 0 |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | 5 |
| 9 | Command | uint8 | See below |

**Command**: Type of power down to perform. The valid values are:

0 = Normal Shutdown

1 = Factory Default: mark applicable NVM parameters as needing to be reset to their default values and shutdown

## Power Down Response

The module sends this response to the Host to acknowledge a power down sequence has started.

Message Usage: Shut down Module System

Message Direction: Sent from Module to Host

Minimum **NumBytes**: 9

Maximum **NumBytes**: 45

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | 0 |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | 0x8005 |
| 8+ | Error/Status Codes: See below. | uint8  +  uint32 | See below |

**Error/Status Codes**:

Indicates status of the Motors and their coordinates, the temperature and status of the tool, and any error codes the module found while processing previous commands.

Values will be 4 byte values, prefaced by a byte identifier to indicate what type of problem is being discussed.

# Status Messages

The Module sends status messages to the Host to inform it of various Module statuses:

* Some statuses are essential for implementation of all Production products, so the corresponding status dispatches are automatically enabled at startup and cannot be disabled.
* Some statuses are only needed in an Test environment, so the Test Host is allowed to enable and disable the corresponding status dispatches as needed.
* Some Engineering statuses change at a high rate. It is assumed that the Test Host is capable of receiving all messages without loss and that acknowledgement of certain high-frequency Engineering messages, and subsequent resends of unacknowledged messages by the Module, is unnecessary.

## Now Printing Dispatch

The module will send a Now Printing Dispatch to the Host at a ~1 second frequency to inform the Host of the motors, tool, and other environmental variables’ statuses. The Module shall give the step count that each motor (x, y, z) is currently at, the temperature reading on the tool (if applicable), and the current system time.

Message Usage: Give printer status update

Message Direction: Sent from Module, Sent to Host

Response: Generic Response from Host

Response Timeout: 1 second

Minimum **NumBytes**: 10

Maximum **NumBytes**: 10

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | Incremental |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | 0xA |
| 9-10 | X motor current coordinates | uint16 | See below |
| 11-12 | Y motor current coordinates | uint16 | See below |
| 13-14 | Z motor current coordinates | uint16 | See below |
| 15-18 | Time of the Day | uint32 | See below |

**Motor coordinates**:

The Module will determine the maximum and minimum step count of each of its available axis upon initialization. The current value of the step count is what shall be passed as a coordinate.

**Time of Day:**

Byte 15 : seconds

Byte 16 : minutes

Byte 17: hours

## Time Dispatch

The Module will become synchronized with Host’s time and then subsequently, the Module will continue to send Time of the Day to the Host in other dispatches.

This message is automatically enabled and cannot be disabled by the Host.

The Module sends the Generic Response to the Host to confirm receipt of the Time Dispatch.

Message Direction: Sent from Host

Response: Generic Response

Response Timeout: 1 sec

Minimum **NumBytes**: 12

Maximum **NumBytes**: 64

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumBytes | uint16 | 9 |
| 3 | TransactionID | uint8 | Incremental |
| 4-5 | Checksum | uint16 | See Section 4.6 |
| 6 | CmdType | uint8 | See Section 4.4 |
| 7-8 | CmdID | uint16 | 0xB |
| 7 | Seconds | uint8 | See below |
| 8 | Minutes | uint8 | See below |
| 9 | Hour | uint8 | See below |
| 10 | Am/Pm | uint8 | See below |

**Seconds:** Current seconds of the host. The range of values is 0 to 59.

**Minute**: Current minute of the host. The range of values is 0 to 59.

**Hour**: Current hour of the host. The range of values is 0 to 23.

**AM/PM:** Current hour frame of the day, day or night. The range is 0 or 1, 0 = AM.

## System Status Dispatch

The Module sends the System Status Dispatch to report the various statuses always needed for User display or other routine operational needs. The Module shall send this message once at startup, that is, after the Module Configure Response is sent to the Host. Thereafter, the message is sent automatically whenever any of the fields contained in the message changes.

This message is automatically enabled and cannot be disabled by the Host.

The Host sends the Generic Response to the Module to confirm receipt of the System Status Dispatch.

Message Usage: Production Infotainment System

Message Direction: Sent from Module

Response: Generic Response

Response Timeout: 500 msec

Test: Full validation by Test Team

Minimum **NumMsgBytes**: 12

Maximum **NumMsgBytes**: 256

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 12 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 12 |
| 7 | SummaryStatus | uint8 | See below |
| 8-9 | ScannableUnplayedTracks | uint16 | See below |
| 10-11 | TuneMixStatus | uint16 | See below |

**SummaryStatus**: Reports the Module’s summary status, with bit fields defined as:

| **Bit7** | **Bit 6** | **Bits 5** | **Bit 4** | **Bits 3-2** | **Bits 1-0** |
| --- | --- | --- | --- | --- | --- |
| AudioPresent | AdPending | AdImminent | HiTemp | AntennaStatus | NumBars |

AudioPresent: Audio present status, with values:

0 = Non-zero audio is not being output on the Module’s I2S interface

1 = Non-zero audio is being output on the Module’s I2S interface

AdPending: Indicates if there is an Ad Insertion or Ad Replacement currently scheduled, with values:

0 = no Ad Insertion or Ad Replacement is scheduled

1 = Ad Insertion or Ad Replacement is scheduled

AdImminent: Indicates that an Ad Insertion or Ad Replacement is expected to happen within the next second, with values:

0 = Ad Insertion or Ad Replacement is not imminent

1 = Ad Insertion or Ad Replacement is imminent

When the Ad Replacement feature is used, AdImminent is only set when an new ad replacement session is imminent. AdImminent will not be set if an existing ad replacement session is re-played.

HiTemp: Reports the Module’s temperature warning status, with values:

0 = the Module’s temperature is below the warning temperature threshold

1 = the Module’s temperature is at or above the warning temperature threshold

NumBars: Reports the Module’s summary reception status, with values:

0 = 0 bars – very weak or no signal

1 = 1 bar – weak signal

2 = 2 bars – marginal signal

3 = 3 bars – good signal

AntennaStatus: Reports the antenna status, with values:

0 = Antenna okay

1 = Antenna disconnected

2 = Antenna shorted

**ScannableUnplayedTracks**: Indicates if there are enough scannable tracks available (i.e., two or more) to initiate TuneScan. The valid values are:

0 = Insufficient scannable tracks are available to initiate TuneScan

1 = Sufficient scannable tracks are available to initiate TuneScan

**TuneMixStatus**: Summary status for TuneMix, with bit fields defined as:

| **Bit 15** | **Bit 14** | **Bits 13-6** | **Bits 5-0** |
| --- | --- | --- | --- |
| Playable | HasStart | Reserved | NumComponents |

NumComponents: The number of valid, available component channels assigned to TuneMix. The valid values are 0-24.

HasStart: Indicates if the Module has the start of at least one track from one of TuneMix’s component channels:

0 = does not have start of any tracks

1 = has start for at least one track

Playable: Indicates if the Host can select TuneMix for playback:

0 = TuneMix is not playable (i.e., does not have sufficient component music channels)

1 = TuneMix is playable

## Status Monitor Dispatch

The Host sends the Status Monitor Dispatch to the Module to control the reporting of specified Module statuses. On startup, all status monitors that can be enabled/disabled are disabled.

When a status message becomes enabled, the Module will respond with the current state of the status item in an appropriate Dispatch. Thereafter, the Module will check the status item periodically and if the value has changed, in general the Module will send an appropriate Dispatch with the new value of the status item(s).

When the Module receives a Status Monitor Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, and performs any processing associated with that command. If these operations are successful then the Module sends the Generic Response to the Host. If these operations are not successful then the Module sends the Status Monitor Error Response to the Host.

Message Usage: Production Infotainment System

Message Direction: Sent to Module

Response: Generic Response or Status Monitor Error Response

Response Timeout: 500 msec

Test: Full validation by Test Team

Minimum **NumMsgBytes**: 9

Maximum **NumMsgBytes**: 64

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 9 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 15 |
| 7 | Command | uint8 | See below |
| 8 | StatusMessage | uint8 | See below |

**Command**: Action to perform. The valid values are:

0 = Disable the sending of the specified status message

1 = Enable the sending of the specified status message

2 = Send the specified status message once (i.e., one-shot), ignore any response, and do not send retries if no response is received

3 = Reset any latched status associated with the specified status message

**StatusMessage**: Status message to perform the requested action on. The valid values are:

Production Host dispatches, valid for **Command**=one-shot:

0 = Time Dispatch

1 = System Status Dispatch

2 = Subscription Status Dispatch

3 = Authorization Status Dispatch

4 = Channel Status Dispatch

5 = Event Report Dispatches (one-shot sends last 10 events)

Production Host dispatches, valid for **Command**=disable, enable, one-shot:

40 = Antenna Aiming Dispatch

41 = Signal Status Summary Dispatch

Production Host dispatches, valid for **Command**=disable, enable, one-shot, reset:

100 = Messaging Status Dispatch

Engineering Only dispatches, for **Command**=disable, enable, one-shot, reset:

101 = Queue Status Dispatch

102 = Audio Decoder Status Dispatch

103 = Audio Decoder Detailed Status Dispatch

104 = SPI Status Dispatch

105 = Authorization Detailed Status Dispatch

106 = Application Payload Element Dispatch

107 = Channel Info Dispatch

108 = BIC Status Dispatch

109 = CAP Status Dispatch

110 = DMA Status Dispatch

111 = CA debug dispatches (CA Error Indication Dispatch, CA Debug Indication Dispatch, and CA Diagnostic Indication Dispatch)

120 = AVS Status Dispatch

140 = ADI, NXP, or Broadcom Tuner Status Dispatch

150 = C/N Status Dispatch

160 = High Band MFC Dispatch

161 = High Band Signal Quality Dispatch

162 = High Band ATE Dispatch

163 = Service Layer Status Dispatch

164 = High Band FFT Dispatch

165 = High Band Overlay Pipe Status Dispatch

180 = Low Band MFC Dispatch

181 = Low Band Signal Quality Dispatch

182 = Low Band ATE Dispatch

184 = Low Band Overlay Pipe Status Dispatch

185 = Low Band Pipe Header Dispatch

186 = High Band Fast C/N Dispatch

Engineering Only dispatches, for **Command**=disable, enable:200 = TuneMix Track Evaluation Dispatch

201 = Ad Status Dispatch

## Status Monitor Error Response

When the Module receives a Status Monitor Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, and performs any processing associated with that command. If these operations are not successful then the Module sends the Status Monitor Error Response to the Host.

Message Usage: Production Infotainment System

Message Direction: Sent from Module

Test: Full validation by Test Team

Minimum **NumMsgBytes**: 56

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 56 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 16 |
| 7 | ErrorCode | uint8 | See below |
| 8-55 | ErrorText[48] | uint8 | See below |

**ErrorCode**: Indicates why the command failed. The valid values are:

0 = requested **Command** is unknown; message ignored

2 = requested **StatusMessage** is unknown; message ignored

3 = requested **StatusMessage** is not valid for the specified **Command**; message ignored

**ErrorText**: ASCII characters describing the error in human-readable form. Any unused bytes at the end of the text will be set to 0. The Host should ignore the content in this field as it is for Engineering Use Only and could be changed at any time.

# Firmware Update Messages

Firmware Update (FWU) is the process of installing new firmware (i.e., executable code or data) into the Module. The Host manages FWU by breaking a binary file into packets and sending them sequentially to the Module. The binary file contains all of the information needed to identify what it is, what its version is, etc., and define its interdependencies with other Gen8 firmware. The Module verifies the validity of the individual packets and the overall binary file, then programs its nonvolatile memory with this new firmware. A Module reset is required before the new firmware will take effect.

## FWU Start Dispatch

The Host sends the FWU Start Dipatch to the Module to begin the FWU process. Only one FWU can be in progress at any time, so if the Module receives this message and FWU is in progress then it is terminated and FWU is restarted using the new command data. Once the Host receives a successful response it can immediately begin sending FWU Packet Dispatches to the Module to download the new binary.

The FWU process must be followed for each binary (i.e., COP or CAP) to be flashed. The Host has the following options for updating the Module's firmware on any given power cycle:

* Only update the COP firmware.
* Only update the CAP firmware.
* Update the COP and CAP firmware as a non-coupled update.
* Update the COP and CAP firmware as a coupled update.

If a coupled COP and CAP are to be flashed, then both must be specified as being flashed as coupled, and only if both are successfully flashed prior to a power cycle will the Module use them on the next power cycle.

If a non-coupled COP and CAP are to be loaded, then whatever binary(ies) are successfully flashed will be used on the next power cycle.

Message Usage: Production Infotainment System

Message Direction: Sent to Module

Response: FWU Start Response or FWU Start Error Response

Response Timeout: 500 msec

Test: Full validation by Test Team

Minimum **NumMsgBytes**: 11

The **Control** field was a late addition to this message. To retain backward compatibility, the Module must allow for FWU Start Dispatch to contain only 11 bytes, and use a default value for **Control** (i.e., 0).

Maximum **NumMsgBytes**: 256

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 12 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 44 |
| 7-10 | NumBinaryBytes | uint32 | See below |
| 11 | Control | uint8 | See below |

**NumBinaryBytes**: Total number of bytes in the binary to be downloaded. The valid values are 1024 - 1,638,400.

**Control**: Bitfield of options for controlling FWU download and processing.

|  |  |
| --- | --- |
| **Bits 7-1** | **Bit 0** |
| Reserved | Coupled |

Coupled: Indicates whether or not new COP and CAP firmware must both be programmed successfully before either can be used. E.g., this would be done in the case where a change to the COP/CAP interface requires the firmware for both processors to be updated at the same time to ensure compatibility. The value values are 0=not coupled, 1=coupled.

## FWU Start Response

When the Module receives a FWU Start Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, and performs any processing associated with that command. If these operations are successful then the Module sends the FWU Start Response to the Host.

Message Usage: Production Infotainment System

Message Direction: Sent from Module

Test: Full validation by Test Team

Minimum **NumMsgBytes**: 8

Maximum **NumMsgBytes**: 32

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 8 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 45 |
| 7 | Status | uint8 | See below |

**Status**: Provides status on the FWU process. The valid values are:

0 = FWU process successfully started.

1 = Previous FWU was in progress and has been aborted. FWU process successfully started.

## FWU Start Error Response

When the Module receives a FWU Start Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, and performs any processing associated with that command. If these operations are not successful then the Module sends the FWU Start Error Response to the Host.

Message Usage: Production Infotainment System

Message Direction: Sent from Module

Test: Full validation by Test Team

Minimum **NumMsgBytes**: 56

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 56 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 46 |
| 7 | ErrorCode | uint8 | See below |
| 8-55 | ErrorText[48] | uint8 | See below |

**ErrorCode**: Indicates why the command failed. The valid values are:

0 = too few bytes specified for the binary length

1 = too many bytes specified for the binary length

2 = too few bytes specified for the binary length; FWU that was in progress has been terminated

3 = too many bytes specified for the binary length; FWU that was in progress has been terminated

4 = a binary has already been flashed as coupled, cannot load a non-coupled binary

5 = a binary has already been flashed as non-coupled, cannot load a coupled binary

6 = a binary has already been flashed as coupled, cannot load a non-coupled binary; FWU that was in progress has been terminated

7 = a binary has already been flashed as non-coupled, cannot load a coupled binary; FWU that was in progress has been terminated

**ErrorText**: ASCII characters describing the error in human-readable form. Any unused bytes at the end of the text will be set to 0. The Host should ignore the content in this field as it is for Engineering Use Only and could be changed at any time.

## FWU Packet Dispatch

The binary file being downloaded is broken into 1024-byte snippets by the Host. The Host sends the FWU Packet Dispatch containing the first of those snippets to the Module. The Module responds with a FWU Packet Response confirming that snippet has been received. The Host then sends the next snippet, and so on, until the entire file has been transferred. The Host must send the snippets in sequential order.

Message Usage: Production Infotainment System

Message Direction: Sent to Module

Response: FWU Packet Response or FWU Packet Error Response

Response Timeout: 500 msec

Test: Full validation by Test Team

Minimum **NumMsgBytes**: 1035

Maximum **NumMsgBytes**: 1400

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 1035 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 47 |
| 7-10 | PacketID | uint32 | See below |
| 11-1034 | PacketData[1024] | uint8 | See below |

**PacketID**: Sequence number of the current packet. The starting value of this field is 0 and it must increment by 1 for each subsequent packet. The valid range is all unsigned integer values, though in reality it will be limited by the size of the largest allowable binary file.

**PacketData**: Binary data (snippet). This field will always be 1024 bytes long. For the last packet of the download, the last (1024 – (*NumBinaryBytes* from the FWU Start Dispatch modulo 1024)) bytes are invalid and will be ignored by the Module. *PacketData* must contain at least one byte of valid data.

## FWU Packet Response

When the Module receives a FWU Packet Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, and performs any processing associated with that command. If these operations are successful then the Module sends the FWU Packet Response to the Host.

Message Usage: Production Infotainment System

Message Direction: Sent from Module

Test: Full validation by Test Team

Minimum **NumMsgBytes**: 12

Maximum **NumMsgBytes**: 32

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 12 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 48 |
| 7 | Status | uint8 | See below |
| 8-11 | PacketID | uint32 | See below |

**Status**: Provides status on the FWU packet. The valid values are:

0 = Packet accepted. Ready for next packet.

1 = Packet accepted. No more packets are expected (i.e., this was the last packet).

**PacketID**: Sequence number of the last packet that was successfully processed.

## FWU Packet Error Response

When the Module receives a FWU Packet Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, and performs any processing associated with that command. If these operations are not successful then the Module sends the FWU Packet Error Response to the Host.

Message Usage: Production Infotainment System

Message Direction: Sent from Module

Test: Full validation by Test Team

Minimum **NumMsgBytes**: 60

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 60 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 49 |
| 7 | ErrorCode | uint8 | See below |
| 8-11 | PacketID | uint32 | See below |
| 12-59 | ErrorText[48] | uint8 | See below |

**ErrorCode**: Indicates why the command failed. The valid values are:

0 = FWU is not in progress so this command is invalid.

1 = Packet ignored. The first *PacketID* after the start of FWU was not 0.

2 = Packet ignored. The *PacketID* was out of sequence. *PacketID’s* must be monotonically increasing.

3 = Packet ignored. A packet was received after all expected packets had been processed.

4 = FWU aborted. Invalid binary header start bytes.

5 = FWU aborted. Invalid binary header end bytes.

6 = FWU aborted. Unknown binary type specified in binary header.

7 = FWU aborted. Invalid binary header number of bytes.

8 = FWU aborted. Binary header number of bytes does not agree with FWU Start Dispatch.

9 = FWU aborted. Invalid binary header checksum.

10 = FWU aborted. Invalid Firmware checksum.

11 = FWU aborted. Attempt to load the same FW without resetting Module.

12 = FWU aborted. Invalid binary header version.

13 = FWU aborted. Invalid binary header variable field.

14 = FWU aborted. Digital signature is missing from binary header or is invalid.

16 = FWU aborted. Invalid number of header bytes specified in binary header.

17 = FWU aborted. Attempt to revert to a previous version of COP FW.

18 = FWU aborted. Maximum number of attempts for this binary has been exceeded.

**PacketID**: Sequence number of the last packet that was successfully processed. If no packet has been successfully processed then PacketID is set to 0xFFFFFFFF.

**ErrorText**: ASCII characters describing the error in human-readable form. Any unused bytes at the end of the text will be set to 0. The Host should ignore the content in this field as it is for Engineering Use Only and could be changed at any time.

## FWU Status Dispatch

When FWU is in progress the Module sends the FWU Status Dispatch to the Host to report the status of the FWU process. The Module will send this indication whenever any of the FWU statuses contained in the message changes, but will send it no more often than once per second.

The Host sends the Generic Response to the Module to confirm receipt of the FWU Status Dispatch.

Message Usage: Production Infotainment System

Message Direction: Sent from Module

Response: Generic Response

Response Timeout: 500 msec

Test: Full validation by Test Team

Minimum **NumMsgBytes**: 28

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 28 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 50 |
| 7 | State | uint8 | See below |
| 8-11 | NumBytesRcvd | uint32 | See below |
| 12-15 | NumBytesProcessed | uint32 | See below |
| 16-17 | EstSecondsRemaining | uint16 | See below |
| 18-19 | NumNVMEraseFailures | uint16 | See below |
| 20-21 | NumNVMWriteFailures | uint16 | See below |
| 22-23 | NumNVMReadFailures | uint16 | See below |
| 24-25 | NumReadbackBytesMismatch | uint16 | See below |
| 26-27 | NumOutOfOrderPackets | uint16 | See below |

**State**: Indicates the state and status of the FWU process. The valid values are:

0 = FWU is in progress

1 = FWU completed successfully

2 = FWU aborted: NVM erase failed

3 = FWU aborted: NVM write failed

4 = FWU aborted: NVM read failed

5 = FWU aborted: NVM readback verification failed

6 = FWU aborted: Digital signature verification failed

**NumBytesRcvd**: Number of valid FWU data bytes the Module has received. When this number is equal to the number of bytes specified in the FWU Start Dispatch all data will have been received by the Module and the download phase of FWU is complete. The valid values are 0 thru *NumBinaryBytes* specified in the most recently received FWU Start Dispatch.

**NumBytesProcessed**: Number of FWU data bytes the Module has processed and written to its nonvolatile memory, if necessary. When this number is equal to the number of data bytes specified in the FWU Start Dispatch the binary file will have been completely stored in the Module’s nonvolatile memory. However, since the Module has to take other actions to complete the FWU this does not mean FWU is complete. The valid values are 0 thru *NumBytesRcvd*.

**EstSecondsRemaining**: Estimated number of seconds remaining until FWU is complete, assuming the Host provides the binary data to the Module faster than programming occurs. This number will take into account flash erase time, programming time, binary verification time, etc. The valid range is all unsigned integer values, though in reality it will be limited by the size of the largest allowable binary file. LSB = 1 sec.

**NumNVMEraseFailures**: Number of NVM erase commands that failed during the current FWU attempt.

**NumNVMWriteFailures**: Number of NVM write commands that failed during the current FWU attempt.

**NumNVMReadFailures**: Number of NVM read commands that failed during the current FWU attempt.

**NumReadbackBytesMismatch**: Number of bytes read back from NVM that did not match the binary that was sent to the Module for programming for the current FWU attempt.

**NumOutOfOrderPackets**: Number of FWU Packet Dispatches received that did not have the *PacketID* that the Module expected.

# Engineering Messages – Memory

The messages in the chapter are intended for use in an Engineering and/or Test environment. These messages will be disabled in production Module Firmware and should not be used by a production Host.

If the definition of an Engineering message changes, then an attempt will be made to make it backward and forward compatible with the message’s previous definition. However, if this is not doable without unnecessarily complicating the message, then compatibility will not be preserved.

## Write Memory Dispatch

The Host sends the Write Memory Dispatch to the Module to write a value to a specific memory address in the COP’s address space. This could include DDR, ITCM , DTCM, or memory mapped hardware registers.

There is nothing to prevent this command from overwriting code, data, vital hardware registers, etc. It is up to the Host to know what it is writing and how it will affect Module Firmware operation. Given the danger inherent in this message, it is recommended that only Engineering tools implement this message and that Production Hosts do not have access to this message.

When the Module receives a Write Memory Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, and performs any processing associated with that command. If these operations are successful then the Module sends the Generic Response to the Host. If these operations are not successful then the Module sends the Write Memory Error Response to the Host.

Message Usage: SiriusXM Engineering

Message Direction: Sent to Module

Response: Generic Response or Write Memory Error Response

Response Timeout: 500 msec

Test: General message validation by Test Team; data verification by Development Team

Minimum **NumMsgBytes**: 20

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 20 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 108 |
| 7-11 | IVSMID[5] | uint8 | See below |
| 12-15 | Address | uint32 | See below |
| 16-19 | Data | uint32 | See below |

**IVSMID**: This is the IVSMID reported to the Host in the Module Configure Response. Because of the criticality of this message, this field is included to help prevent a Host from accidentally sending this message to the Module.

**Address**: Address of memory to write to. The valid range is 0x00000000 through 0xFFFFFFFC, and must be evenly divisible by 4. The sender of the message is responsible for verifying that the address is a valid, writable memory location and will not cause any undesirable side effects.

**Data**: Value to write to the specified address.

## Write Memory Error Response

When the Module receives a Write Memory Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, and performs any processing associated with that command. If these operations are not successful then the Module sends the Write Memory Error Response to the Host.

Message Usage: SiriusXM Engineering

Message Direction: Sent from Module

Test: General message validation by Test Team; data verification by Development Team

Minimum **NumMsgBytes**: 56

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 56 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 109 |
| 7 | ErrorCode | uint8 | See below |
| 8-55 | ErrorText[48] | uint8 | See below |

**ErrorCode**: Indicates why the command failed. The valid values are:

0 = invalid **IVSMID** – data not written

1 = **Address** is not evenly divisible by 4 – data not written

**ErrorText**: ASCII characters describing the error in human-readable form. Any unused bytes at the end of the text will be set to 0. The Host should ignore the content in this field as it is for Engineering Use Only and could be changed at any time.

## Read Memory Dispatch

The Host sends the Read Memory Dispatch to the Module to read a value from a specific memory address in the COP’s address space. This could include DDR, ITCM , DTCM, or memory mapped hardware registers.

There is nothing to prevent this command from reading from an invalid address, or cause fatal effects if reading from certain hardware registers. It is up to the Host to know what it is reading and how it will affect Module Firmware operation. Given the danger inherent in this message, it is recommended that only Engineering tools implement this message and that Production Hosts do not have access to this message.

When the Module receives a Read Memory Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, performs any processing associated with that command, and sends the Read Memory Response to the Host.

Message Usage: SiriusXM Engineering

Message Direction: Sent to Module

Response: Read Memory Response

Response Timeout: 500 msec

Test: General message validation by Test Team; data verification by Development Team

Minimum **NumMsgBytes**: 11

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 11 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 112 |
| 7-10 | Address | uint32 | See below |

**Address**: Address of memory to read from. The valid range is 0x00000000 through 0xFFFFFFFC, and must be evenly divisible by 4. The sender of the message is responsible for verifying that the address is a valid, readable memory location and will not cause any undesirable side effects.

## Read Memory Response

When the Module receives a Read Memory Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, performs any processing associated with that command, and sends the Read Memory Response to the Host.

Message Usage: SiriusXM Engineering

Message Direction: Sent from Module

Test: General message validation by Test Team; data verification by Development Team

Minimum **NumMsgBytes**: 12

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 12 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 113 |
| 7 | Status | uint8 | See below |
| 8-11 | Data | uint32 | See below |

**Status**: Indicates if the read request was successful or not, and why. The valid values are:

0 = successful, data is valid

1 = failed, **Address** is not evenly divisible by 4, data is not valid

**Data**: uint32 read from **Address**.

## Write Serial Flash Dispatch

The Host sends the Write Serial Flash Dispatch to the Module to write the specified number of values to the specified sequential addresses in the COP’s serial flash.

There is nothing to prevent this command from overwriting code, NVM data, etc. It is up to the Host to know what it is writing and how it will affect Module Firmware operation. Given the danger inherent in this message, it is recommended that only Engineering tools implement this message and that Production Hosts do not have access to this message.

When the Module receives a Write Serial Flash Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, and performs any processing associated with that command. If these operations are successful then the Module sends the Generic Response to the Host. If these operations are not successful then the Module sends the Write Serial Flash Error Response to the Host.

Message Usage: SiriusXM Engineering Debug releases only

Message Direction: Sent to Module

Response: Generic Response or Write Serial Flash Error Response

Response Timeout: 500 msec

Test: General message validation by Test Team; data verification by Development Team

Minimum **NumMsgBytes**: 82

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 82 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 110 |
| 7-11 | IVSMID[5] | uint8 | See below |
| 12 | Command | uint8 | See below |
| 13-16 | Address | uint32 | See below |
| 17 | NData | uint8 | See below |
| 18-81 | Data[64] | uint8 | See below |

**IVSMID**: This is the IVSMID reported to the Host in the Module Configure Response. Because of the criticality of this message, this field is included to help prevent a Host from accidentally sending this message to the Module.

**Command**: Controls the erase/write operation to be performed by the Module. The valid values are:

0 = do not erase the serial flash sector before writing the values

1 = erase the enclosing serial flash sector before writing the data

**Address**: Beginning address of serial flash to write. The valid range is 0 thru [size of serial flash – 1]. The Module increments the address accordingly for each data value written.

**NData**: Number of uint8 data items in **Data** to write to the serial flash. The valid values are 1-64.

**Data**: Data to write to the serial flash. Unused entries are ignored by the Module.

## Write Serial Flash Error Response

When the Module receives a Write Serial Flash Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, and performs any processing associated with that command. If these operations are not successful then the Module sends the Write Serial Flash Error Response to the Host.

Message Usage: SiriusXM Engineering

Message Direction: Sent from Module

Test: General message validation by Test Team; data verification by Development Team

Minimum **NumMsgBytes**: 56

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 56 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 111 |
| 7 | ErrorCode | uint8 | See below |
| 8-55 | ErrorText[48] | uint8 | See below |

**ErrorCode**: Indicates why the command failed. The valid values are:

0 = invalid **IVSMID** – data not written

1 = invalid **Command** – data not written

2 = invalid **Address** – data not written

3 = invalid **NData** – data not written

4 = **Address**/**NData** traverses a serial flash sector – data not written

5 = NVM erase failed

6 = NVM write failed

7 = readback failed

8 = this feature is disabled in the currently running code

**ErrorText**: ASCII characters describing the error in human-readable form. Any unused bytes at the end of the text will be set to 0. The Host should ignore the content in this field as it is for Engineering Use Only and could be changed at any time.

## Read Serial Flash Dispatch

The Host sends the Read Serial Flash Dispatch to the Module to read the specified number of values from the specified sequential addresses in the COP’s serial flash.

When the Module receives a Read Serial Flash Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, performs any processing associated with that command, and sends the Read Serial Flash Response to the Host.

Message Usage: SiriusXM Engineering

Message Direction: Sent to Module

Response: Read Serial Flash Response

Response Timeout: 500 msec

Test: General message validation by Test Team; data verification by Development Team

Minimum **NumMsgBytes**: 12

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 12 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 114 |
| 7-10 | Address | uint32 | See below |
| 11 | NData | uint8 | See below |

**Address**: Beginning address of serial flash to read. The valid range is 0 thru [size of serial flash – 1]. The Module increments the address accordingly for each data value read.

**NData**: Number of uint8 data items to read from the serial flash. The valid values are 1-64.

## Read Serial Flash Response

When the Module receives a Read Serial Flash Dispatch it validates the command parameters, verifies the Module is in the proper state to receive the command, performs any processing associated with that command, and sends the Read Serial Flash Response to the Host.

Message Usage: SiriusXM Engineering

Message Direction: Sent from Module

Test: General message validation by Test Team; data verification by Development Team

Minimum **NumMsgBytes**: 72

Maximum **NumMsgBytes**: 128

Message Format:

| **Byte #** | **Field Name** | **Data Type** | **Value** |
| --- | --- | --- | --- |
| 0 | Sync | uint8 | See Section 4.1 |
| 1-2 | NumMsgBytes | uint16 | 72 |
| 3-4 | Checksum | uint16 | See Section 4.6 |
| 5 | TransactionID | uint8 | See Section 4.3 |
| 6 | MessageID | uint8 | 115 |
| 7 | Status | uint8 | See below |
| 8-71 | Data[64] | uint8 | See below |

**Status**: Indicates if the read request was successful or not, and why. The valid values are:

0 = successful, data is valid

1 = failed, invalid **Address**, data is not valid

2 = failed, invalid **NData**, data is not valid

**Data**: uint8 values read from the specified serial flash **Address**. The first **NData** values are valid, and any remaining entries are set to 0.