DRAFT

BSR E1.37-4 – 202x

Entertainment Technology - File Transfer Control with Remote Device Management over DMX512 Networks

CP/2017-1024r2a
TG Draft Revision 4.0.053

Notice and Disclaimer

ESTA does not approve, inspect, or certify any installations, procedures, equipment or materials for compliance with codes, recommended practices or standards. Compliance with an ESTA standard or recommended practice is the sole and exclusive responsibility of the manufacturer or provider and is entirely within their control and discretion. Any markings, identification or other claims of compliance do not constitute certification or approval of any type or nature whatsoever by ESTA.

ESTA neither guaranties nor warrants the accuracy or completeness of any information published herein and disclaim liability for any personal injury, property or other damage or injury of any nature whatsoever, whether special, indirect, consequential or compensatory, directly or indirectly resulting from the publication, use of, or reliance on this document.

In issuing and distributing this document, ESTA does not either (a) undertake to render professional or other services for or on behalf of any person or entity, or (b) undertake any duty to any person or entity with respect to this document or its contents. Anyone using this document should rely on their own independent judgment or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstance.

Published By:
Entertainment Services and Technology Association (ESTA)
271 Cadman Plaza PO Box 23200
Brooklyn, NY 11202-3200
USA
Phone: +1-212-244-1505
Email: standards@esta.org

ESTA's Technical Standards Program

The ESTA Technical Standards Program was created to serve the ESTA membership and the entertainment industry in technical standards related matters. The goal of the program is to take a leading role regarding technology and safety within the entertainment industry by creating recommended practices and standards, monitoring standards issues around the world on benefit of our members, and improving communications and safety within the industry. ESTA works closely with the technical standards efforts of other organizations within our industry including ESA, CITT, USITT and VPLT as well as representing the interests of ESTA members to ANSI, UL, ASCE, ICC, and the NFPA. ESTA is an ANSI Accredited Standards Developer.

The Technical Standards Council (TSC) established by ESTA's Board of Directors to oversee and coordinate the Technical Standards Program. Made up of individuals experienced in standards development work from throughout our industry, the Committee approves all projects undertaken and assigns them to the appropriate working group. The Technical Standards Council employs a Technical Standards Manager to coordinate the work of the Committee and its working groups as well as maintaining a "Standards Watch" on behalf of members. Working groups include: Control Protocols, Electrical Power, Event Safety, Floors, Fog and Smoke, Followspot Positions, Photometrics, Rigging, Stage Machinery, and Prop Weapons Safety.

ESTA encourages active participation in the Technical Standards Program. There are several ways to become involved. The easiest way to actively participate is to respond to any of the public reviews advertised on ESTA's Public Review Documents. The next level of participation requires completion of an application to become a working group member; applications are available from the TSP's procedural documents web page. Application as an Observer member affords access to updates on standards development documents. Application as a voting member affords full participation as a consensus voice that helps shape the industry. Requirements for voting membership include responding to letter ballots and attending meetings, but membership in ESTA or any other organization is not a requirement for participation in the TSP. One can also become involved by requesting that the TSC develop a standard or a recommended practice in an area of concern to them.

The Control Protocols Working Group, which authored this standard, consists of a cross section of entertainment industry professionals representing a diversity of interests. ESTA is committed to developing consensus-based standards and recommended practices in an open setting.

Investors in Innovation

The Technical Standard Program (TSP) is financially supported by ESTA and by companies and individuals who make donations to the TSP. Contributing companies and individuals who have helped fund the TSP are recognized as “Investors in Innovation”. These are the Investors in Innovation when this standard was approved by ANSI's Board of Standards Review:

[Insert the current Investors table here. Source from latest Swatch Edition]

Memorial donor: The Estate of Ken Vannice

All donations to the Technical Standards Program benefit the entire program, and are not directed to any specific use or project within the program. Please help support the Technical Standards Program by becoming an Investor in Innovation. Visit our website at http://tsp.esta.org/invest, or contact the ESTA office at 1-212-244-1505 and select "TSP" from the menu.

Contact Information

Technical Standards Manager
Richard J. Nix
ESTA
271 Cadman Plaza PO Box 23200
New York, NY 11202-3200
USA
+1-212-244-1505
richard.nix@esta.org

Technical Standards Council Chairperson
Alan Rowe
I.A.T.S.E Local 728
Phone: 1 310-702-2909
amrowe@iatse728.org

Control Protocols Working Group Co-chairpersons

Milton Davis
Doug Fleenor Design, Inc.
+1-805-481-9599
milton@dfd.com

Javid Butler

Maya Nigrosh

Acknowledgments

The Control Protocols Working Group members, when this document was approved by the working group on [insert WG approval date here], are shown below.

Voting members:

Observer (non-voting) members:

Interest category codes:
CP = custom-market producer DE = designer
DR = dealer rental company G = general interest
MP = mass-market producer U = user

Table of Contents

List of Tables

1. Introduction (Informative)

1.1 Approach taken in this standard

While manufacturers have been able to implement their own proprietary means of file transfer these are limited to supporting transfers from a single manufacturer's Controllers or testing tools to the same manufacturer's devices; this frequently required disconnecting the devices from the existing [DMX] infrastructure. There is no standardized method to support file transfer between a Controller from Manufacturer-A and a device from Manufacturer-B.

This document provides developers of RDM enabled products with a standard method for transfer of files between RDM capable products using the existing basic communication structure provided by [RDM] and its related standards including ANSI E1.33 [RDMnet].

The design approach is intended to facilitate data transfers to Responders that have very limited memory resources as well as devices that can support the largest possible [RDM] packet all while maintaining the physical [DMX] connections to multiple devices from multiple manufacturers.

The process described in this document is referred to as File Transfer Control (FTC) which encompasses file uploads and downloads.

2. Scope

This document sets out the minimum requirements for [RDM] Controllers and [RDM] Responders to successfully support File Transfer Control (FTC) between them.

This standard specifically provides a transfer mechanism that can be applied to memory limited Responders with transmit and receive buffers as small as 64 bytes, while allowing scalable improvements (a net reduction) in transfer times for Responders that have larger buffers.

The extent of Responder normal functionality during the FTC process is not defined by this standard and is outside the scope of this document. This standard includes a method for the Responder to declare whether or not functionality will be limited during the FTC process.

> *The Parameter Messages described in this standard, whilst primarily intended for Firmware updates, may also be used to support the transfer and restoration of other data such as configuration settings, calibration tables, dimmer curves and log files.*

2.1 Source Media

The choice of media support for source data is outside the scope of this standard. Equipment manufacturers are free to choose whatever means they feel is appropriate to their product. Examples of common source data media include USB key, SD Card, Email or Internet download.

2.2 File Structure

A file for transfer should contain data in binary format. A Controller should have no interaction with the content of a file. The Controller should calculate a FileCRC over the contents as outlined in Section 10 of this standard.

This standard places no constraints on the naming of any source file or the internal structure of the file.

Transfer and subsequent installation of firmware files involves inherent risk to the operation of a device.

It is recommended that a method to determine the integrity of the entire transferred firmware file is contained in the firmware file by the files author [*some form of internal Checksum, CRC, hash algorithm, etc.*]. The manner in which this is implemented is the responsibility of the target equipment manufacturer, and beyond the scope of this standard.

3. Definitions

Any definitions given in this standard are particular to this standard and are either not found in a standard dictionary or are used in this standard with a meaning different from what might be found in a standard dictionary.

Upload: An upload refers to the Controller initiated transfer of data from a Controller to a Responder.

Download: A download refers to the Controller initiated transfer of data from a Responder to a Controller.

Controller/Responder: Unless otherwise specified in this standard, the characteristics of an RDM Controller and RDM Responder are as defined in [RDM] Appendix D.

One-to-One: A “One-to-One” transfer is a transfer between a Controller and a single Responder. Each transferred packet is acknowledged..

One-to-Many: A "One-to-Many” transfer is a transfer between a Controller and one or more Responders using [RDM] ALL_DEVICES_ID. The use of [RDM] ALL_DEVICES_ID precludes acknowledgement of the transferred packet.

Cyclic Redundancy Check (CRC): A **cyclic redundancy check** (**CRC**) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to digital data. Blocks of data entering these systems get a short *check value* attached, based on the remainder of a polynomial division of their contents. On retrieval, the calculation is repeated and, in the event the check values do not match, corrective action can be taken against data corruption.

Bootloader: A bootloader is a piece of software that runs at power up before switching to the main application. A bootloader may provide support for FTC Parameter messages in order to transfer and verify the integrity of new firmware. In other cases it may provide for re-flashing of the application code from a previously transferred and validated file.

Normal Function: Responder is operating with the typical features based on the current personality settings.

Fatal Error: Responder is no longer able to resume normal function.

4. Normative References

[DMX] ANSI E1.11-2024
Entertainment Technology -- USITT DMX512-A --
Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories.

Entertainment Services and Technology Association (ESTA)
P.O. Box 23200
Brooklyn, NY 11202-3200
USA
Phone: 1-212-244-1505
Email: standards@esta.org
http://tsp.esta.org

[RDM] E1.20 - xxxx, Entertainment Technology-RDM-Remote Device Management over USITT DMX512 Networks

This standard is maintained by ESTA.

[RDMnet] ANSI E1.33 Entertainment Technology - (RDMnet) Message Transport and Management for ANSI E1.20 (RDM) compatible and similar devices over IP Networks

This standard is maintained by ESTA.

[PIDS-1] ANSI E1.37-1-2011 (R2022)
Entertainment Technology - Additional Message Sets for ANSI E1.20 (RDM) – Part 1, Get/Set Parameter Messages

This standard is maintained by ESTA.

[UTF-8] The Unicode Consortium. The Unicode Standard, Version 13.0.0, (Mountain View, CA: The Unicode Consortium, 2020. ISBN 978-1-936213-26-9)
http://www.unicode.org/versions/Unicode13.0.0/
[EUI] Guidelines for use of a 48-bit Extended Unique Identifier (EUI-48)
http://standards.ieee.org/

This document is maintained by: IEEE Operations Center, 445 Hoes Lane, Piscataway, NJ 08854-4141, USA

[Modbus] Modicon Modbus Protocol Reference Guide
Modbus Organization, Inc.
PI-MBUS-300 Rev J., June 1996

MODICON, Inc., Industrial Automation Systems
One High Street North Andover, Massachusetts 01845

http://modbus.org/docs/PI_MBUS_300

5. Overview (Informative)

This section summarizes FTC operation. It is provided so that readers of the standard can determine suitability of this standard to their requirements.

A FTC Upload (File upload from Controller to Responder) transfer comprises three stages: Initiate, Transfer, and Commit.

Upload	***File Transfer Sequence***
Initiate	"Get Initiate": A Controller obtains details about required transfer configuration information from Responder(s) [optional]
	"Set Initiate": A Controller initiates a FTC Upload
Packet 1 Transfer	1st packet of data is sent based on transfer requirements obtained during Initiate or already known
	Responder verifies integrity of the 1st packet using PacketCRC
Transfer/Verify	Additional packets are transferred and verified
Packet *n* Transfer	*n*th packet of data is sent where *n* is the total number of packets required to complete the transfer based on the file size and the transfer requirements obtained during Initiate (or already known)
	Responder verifies integrity of *n*th packet using PacketCRC and validates the complete file using FileCRC
	Responder verifies that the file is of an expected type and meets manufacturers requirements and integrity check
Commit	Get Commit: Controller requests Responder validation status Set Commit: Controller instructs the Responder to commit (save) the file

This standard introduces additional Parameter Messages ([RDM], Section 10).

Four mandated Parameter Messages are used to control FTC Upload and are the only Messages required to support FTC Upload. Additional optional Parameter Messages can be used to support FTC Downloads and identify the availability of multiple files.

5.1 File Compatibility

It is recommended that the information to determine that the data is suitable for the target Responder is contained in the first few file packets transferred. This might include information such as make, model and version or other details as chosen by the Responder manufacturer.

The manner in which this is determined is the responsibility of the Responder manufacturer, and beyond the scope of this standard.

Manufacturers who elect to encode this information into the first 16 bytes of their files could allow a Responder [from said Manufacturer] to reject the Initiate stage of the transfer should the proprietary encoding not be satisfied.

This standard does not specify the format of any file generated by a manufacturer for a particular Responder.

5.2 Data Integrity

A Cyclic Redundancy Check (CRC) is calculated by the Controller and offered to the Responder for each Transfer packet and for the entire file.

Manufacturers may also wish to embed segment numbers, checksum or CRC information in the content that is transferred over the data link using this standard. The manner in which such checks are embedded in manufacturer's files is beyond the scope of this standard.

5.3 Responder Design Choices

Responder hardware will typically have some design choices that determine how much data may be accepted before it can be saved to storage. Microcontrollers or other off-chip memory devices may impose a "page" size, at which point a finite time (longer than the normal inter-packet time) is required to save data to non-volatile storage. During this time the Responder may not be able to accept RDM messages, or perform normal operations.

Responders may also have memory limitations that limit the size of each RDM packet that can be received.

Such characteristics can be declared so that a Controller can modify it's behaviour appropriately.

5.4 Use of ACK_TIMER

This standard has been specifically designed to avoid the use of [RDM] ACK_TIMER in responses to the FTC_INITIATE, FTC_TRANSFER_UPLOAD, FTC_TRANSFER_DOWNLOAD, FTC_COMMIT and FTC_CANCEL Parameter Messages.

This standard provides an alternate means of requesting delays between GET/SET FTC Parameter messages, with the exception of FTC_FILELIST. The avoidance of the use of ACK_TIMER allows Responders implementing this standard to avoid having to support [RDM] QUEUED_MESSAGES.

5.5 Use of Proxy Devices

Consideration has been given to the use of Proxy Devices ([RDM] Section 8) such as wireless DMX512/RDM data links.

Methods of determining the existence of Proxy Devices are described in [RDM] Section 8.2 and [RDM] Section 8.4.

Controllers using this protocol should expect Proxy Devices to use the ACK_TIMER method described in the [RDM] standard when it is not possible to provide an immediate response to a Controller.

5.6 Cancellation of FTC Process

A Controller may cancel the FTC process at any stage before a SET:FTC_COMMIT.

Receipt by a Responder of a SET:FTC_CANCEL Parameter Message instructs the Responder to abandon all further expectation of data transfer until a new session is initiated by a Controller.

A Responder may indicate an error condition to the Controller through the ResponseStatus field.

Based on the ResponseStatus, a Controller may cancel the current transfer session. Controller designers should note that some ResponseStatus conditions may already have resulted in the Responder abandoning the current transfer.

5.7 Transfer Times

An example calculation of transfer times is provided in Appendix C.

5.8 Limitations

It should be recognized that:

- This standard does not support Sub-devices on Responders. All FTC transactions shall interact with the Responder's Root Sub-device (0x0000).
- Firmware upload is by its nature a maintenance operation and should not normally be undertaken during show performance situations.
- Responders may not be able to provide normal functionality while processing FTC requests and are not required by this standard to do so. Such limitations shall be declared by the Responder Capabilities declaration.
- Controllers may cease or delay generation of NULL start code or other ASC packets during FTC processing.
- This standard does not support the simultaneous transfer of multiple files to an individual Responder.
- Some Responders cannot maintain important settings, calibration tables, etc. during a firmware upgrade.

5.9 Test Mode

Test Mode enables Controllers to instruct a Responder to perform a File Upload without actually committing the file data. This enables the Controller to prove that it has suitable file data for the Responder.

Test Mode may be useful during the development of both Controller and Responder implementations to prove that Responders detect errors in the file data correctly and to verify the calculated FileCRC.

6. Message Structure

All FTC messages shall use the standard RDM message structure as defined in [RDM].

All messages shall return a response type [RDM] ACK with additional information as described in this document unless the message has a parametric RDM error.

6.1 Text Fields

Text fields shall conform to [RDM] Section 10.1.

6.2 Handling of parametric [RDM] errors

The standard NACK responses for Format Error or Data Out of Range, as defined in [RDM], shall be used when the message construct does not comply with this document, unless specifically stated in this document.

The standard NACK response of Unsupported Command Class, as defined in [RDM], shall still apply if the SET_COMMAND is issued for a PID that does not support the SET_COMMAND Class, or a GET_COMMAND is issued for a PID that does not support the GET_COMMAND Class.

7. Controller Requirements

7.1 [RDM] Support Requirements

Controllers implementing this standard shall accept the maximum size RDM packet as defined in [RDM].

7.2 Functional Requirements

Controllers wishing to transfer data using this standard shall determine the required TransferBlock Size, Initial Delay, Inter-Packet Delay Time, Accumulated Byte Count, associated Accumulated Byte Count Delay Time, Validation Delay Time and Communications Offline Delay Time appropriate to a Responder, in accordance with the methods described in this document.

The size of each Transfer packet shall be determined by the Controller from information obtained from the Responder.

Controllers that do not support the GET:FTC_FILELIST Parameter Message will only be able to use this protocol with Responders that support a single FileID unless the FileID is already known to the Controller by some other means.

The handling of Proxy devices is supported in accordance with the methods described in [RDM] Section 8 Proxy Devices.

A Responder may indicate at any point in the transfer that the data offered is not suitable for the Responder. Receipt of such an error through the ResponseStatus field by a Controller shall cause the Controller to abandon further data transfer in the current session.

One-to-Many transfers using [RDM] ALL_DEVICES_ID, [MANUFACTURER Broadcast] , whilst possible, are outside the scope of this document.

The use of [RDM] BROADCAST_ALL_DEVICES_ID is prohibited with the FTC Parameter Messages described in this standard.

8. Responder Requirements

There are some minimum requirements that should be facilitated by Responders to reliably use the transfer scheme detailed in this document. Responder manufacturers shall observe the following points to ensure the integrity of their products during any FTC process.

8.1 Design

Responders should implement a design whereby a transfer can be resumed at any stage of the FTC process in the case of any error condition created by the Controller or by loss of data.

Responder designs should include a recovery method to restore the device in the event of an error where the transfer process and commit cannot be completed.

Responders supporting firmware uploads using FTC may provision for a Bootloader that supports this standard. Such a Bootloader should implement some form of recovery mechanism after any fatal disruption of the FTC process. Discovery in accordance with [RDM] to allow subsequent retry of the FTC Upload may be used to achieve this.

Responder designs may, in the event of failure to complete the FTC process, require an alternate fatal error recovery that requires physical intervention – such as physical reset, USB uploads or solutions not described by this standard. Such devices shall declare this in accordance with Section 8.6.7 of this standard. Controllers may use this declaration to inform the user of the risks associated with the proposed transfer.

Responders that can process One-to-Many transfers using [RDM] ALL_DEVICES_ID shall declare this as outlined in Section 8.6.7 of this standard.

8.2 Responder Firmware

Where possible, Responders should not commit any updated firmware before data transfer has completed and data validation has been confirmed. If a Responder has to use partial commit due to hardware constraints then the Responder shall set the FTC_FAIL_MAY_BRICK bit in the Capabilities Bit Field in accordance with Section 8.6.7 of this standard.

8.3 Responder Bootloader

Responders may wish to switch from normal function to an alternate or “Bootloader” application in order to process FTC Parameter Messages.

Such Responders shall set the FTC_BOOTLOADER_SWITCH bit in the Capabilities Bit Field in accordance with Section 8.6.7 of this standard.

Such Responders shall implement at least the SET:FTC_INITIATE Parameter Message before any switching occurs.

8.4 Responder Limited Behaviour

A Responder may limit or stop its normal functionality during the FTC process.

Responders shall declare if they allow [DMX] refresh or limit normal functions using the Capabilities Bit Field in accordance with Section 8.6.7 of this standard.

> *For example, a Responder may allow [DMX] (Null Start Code) packet processing while transferring a firmware file, but may not respond to non-FTC [RDM] messages.*

8.5 Responder FailSafe Recovery

Responders should allow for the case whereby a Controller may have stopped sending FTC Parameter Messages part way through the FTC process.

> *For example, a test tool may have been used to upload firmware, and removed prior to completion of the FTC Process. Failsafe Recovery ensures that the Responder returns to normal operation.*

Following receipt of an SET:FTC_INITIATE, if no further FTC Parameter Messages have been received after 10 minutes, Responders shall cancel the current session and attempt to revert to normal operation. Responders

that have requested additional time delays using other FTC Parameter Messages shall account for those delays prior to starting the 10 minute Failsafe Recovery timer.

Controllers wishing to delay the FTC_COMMIT may issue a GET:FTC_TRANSFER_UPLOAD to restart the timeout period.

8.6 Responder Declarations

Typically, Responder hardware will have some design constraint(s) that affect the FTC process.

Responders may have memory limitations that limit the size of each RDM packet that can be received.

Responders may be restricted on how much data may be accepted before it must be saved to storage.

Microcontrollers or other off-chip memory devices may impose a “page” size, at which point a finite time (and possibly longer than the normal inter-packet time) is required to save the data to non-volatile storage.

During FTC process the Responder may not be able to accept DMX Packets, accept RDM messages, or perform normal operations.

Responder characteristics and requirements are obtained using the GET/SET:FTC_INITIATE Parameter Messages in accordance with Section 13.1 of this standard.

8.6.1 TransferBlock Size (8-bit):

The Responder shall declare the number of bytes of file data to be sent in each packet.

The maximum allowed TransferBlock Size is restricted by the maximum Parameter Data Length imposed by the [RDM] Standard.

For Upload: The valid range is 0x01-0xE0 (1-224 bytes). A Responder shall not request a TransferBlock Size of 0 bytes.

For Download: The valid range is 0x01-0xDF (1-223 bytes). A Responder shall not offer a TransferBlock Size of 0 bytes.

This value shall be used by a Controller to determine the number of bytes of file data to send to or receive from the Responder in each FTC_TRANSFER_UPLOAD or FTC_TRANSFER_DOWNLOAD Parameter Message packet.

The byte limitation is derived from (i) the 231 byte maximum PDL payload as defined in [RDM], and (ii) the FTC header and PacketCRC within each Transfer packet.

Note that the number of bytes of file data in the last Transfer packet sent may be less than the TransferBlock Size.

8.6.2 Initial Delay Time (32-bit):

Minimum time (in ms) that the Responder requires the Controller to wait after the Initiate stage before the first SET:FTC_TRANSFER_UPLOAD or GET:FTC_TRANSFER_DOWNLOAD Parameter Message is sent.

The valid range is 0x00000000-0x00400000. (Max 4,194,304 ms or approx. 1.2hrs) and a value of 0 represents the minimum allowed time of 176 µs.

This value shall be used by a Controller to determine when to send the first SET:FTC_TRANSFER_UPLOAD or GET:FTC_TRANSFER_DOWNLOAD Parameter Message.

Responder designers are encouraged to keep this Initial Delay time as low as possible, and preferably zero. In any event, successive packets cannot be sent with less than 176 µs inter-packet time as defined in the [RDM] standard.

Controllers in receipt of a value out of range are expected to cancel the process or default to the maximum valid value.

8.6.3 Inter-Packet Delay Time (16-bit):

Minimum time (in ms) that the Responder requires the Controller to wait after each SET:FTC_TRANSFER_UPLOAD before the next SET:FTC_TRANSFER_UPLOAD Parameter Message can be sent.

Minimum time (in ms) that the Responder requires the Controller to wait after each GET:FTC_TRANSFER_DOWNLOAD before the next GET:FTC_TRANSFER_DOWNLOAD Parameter Message can be sent.

The valid range is 0x0000-0xFFFF. (Max 65,535ms, approx. 1 minute) and a value of 0 represents the minimum allowed time of 176 µs.

Responder designers are encouraged to keep this Inter-Packet Delay Time as low as possible, and preferably zero. In any event, successive packets cannot be sent with less than 176us inter-packet time as defined in the [RDM] standard.

8.6.4 Accumulated Byte Count (32-bit):

A Responder may declare any additional delay required after “N” bytes have been transferred to the Responder.

The number of bytes “N” shall be returned in this field. The valid range is 0x00000000-0xFFFFFFFF.

A Responder that does not require any additional delay shall set this field to 0x00000000.

The Accumulated Byte Count shall be used by a Controller to determine when to further delay packet transfers using the Accumulated Byte Delay Time.

8.6.5 Accumulated Byte Count (32-bit):

A Responder that requires additional delay after “N” bytes have been transferred to the Responder shall set the required additional delay (in ms) in this field.

The valid range is 0x0000-0xFFFF. (Max 65,535 ms, approx. 1 min.)

A Responder that does not require any additional delays shall set this field to 0x0000.

A Controller shall use this field to determine the additional time to wait (in ms) once the number of transmitted bytes reaches the declared Accumulated Byte Count.

This allows time for the Responder(s) to save the accumulated data to non-volatile storage or perform other background tasks as may be required by their hardware designs.

Controllers shall reset their internal Byte counter every time the delay is applied.

8.6.6 Validation Delay Time (16-bit):

The valid range is 0x0000-0xFFFF. (Max 65,535 ms, approx. 1 min.)

This time shall be declared by a Responder to indicate how long a Controller should wait (in ms) after receiving the ResponseStatus FTC_RS_TRANSFER_COMPLETE.

For Uploads, this delay indicates how long the Responder requires to complete any Validation stage. This time shall be used by a Controller to wait (in ms) before advancing to the Commit stage and sending a GET/SET:FTC_COMMIT Parameter Message.

For Downloads this delay indicates how long the Responder requires to complete any calculation of a FileCRC. This time shall be used by a Controller to wait (in ms) before sending a GET:FTC_TRANSFER_DOWNLOAD Parameter Message with a command of FTC_TD_GET_FILE_CRC in order to retrieve any Responder calculated FileCRC.

8.6.7 Responder Capabilities (32-bit):

Responder capabilities (i.e., accepts Uploads/Downloads, processes/generates CRCs, supports Test mode, etc.) are reported using a 32-bit Capabilities Bit Field, in Big-Endian order. These are defined in **Error! Reference source not found.**

All unused and/or undefined bits shall be set to (0), Active bits shall be set to (1) unless otherwise specified.

8.6.8 Max Inter-Packet Delay Time (16-bit):

Responders that support the use of [RDM] ALL_DEVICES_ID with FTC Parameter Messages shall set the Max Inter-Packet Delay Time to the worst case Inter-packet Delay time (in ms) [excluding any Accumulated Byte Delay Time] that might be required to process the Transfer packet.

The valid range is 0x0000-0xFFFF. (Max 65,535 ms, approx. 1 min.)

Responders that support the use of the [RDM] ALL_DEVICES_ID cannot inform a Controller about any dynamic Inter-Packet delay requirements via the ResponseStatus mechanism.

Responders that do not support the use of [RDM] ALL_DEVICES_ID with FTC Parameter Messages shall set this field to 0x0000.

Controllers wishing to use [RDM] ALL_DEVICES_ID with FTC Parameter Messages shall use the Max Inter-Packet Delay Time instead of the Inter-Packet Delay Time described in 8.6.3 and shall continue to use the Accumulated Byte Count and Accumulated Byte Delay Time.

8.7 Responder Use of ACK_TIMER

Responders implementing this standard shall reply to the FTC_INITIATE, FTC_TRANSFER_UPLOAD, FTC_TRANSFER_DOWNLOAD, FTC_COMMIT and FTC_CANCEL Parameter Messages without use of the [RDM] ACK_TIMER mechanism.

A Responder shall always ACK these Parameter Messages provided there are no RDM packet format errors in the request. In the event of such low levels errors the existing [RDM] NACK response shall be used.

Responder requiring additional time shall ACK the request and return the required delay using the ResponseStatus and ResponseData fields as described in this standard in Section 12.1 of this standard.

8.8 Responder Handling of Data Out of Range

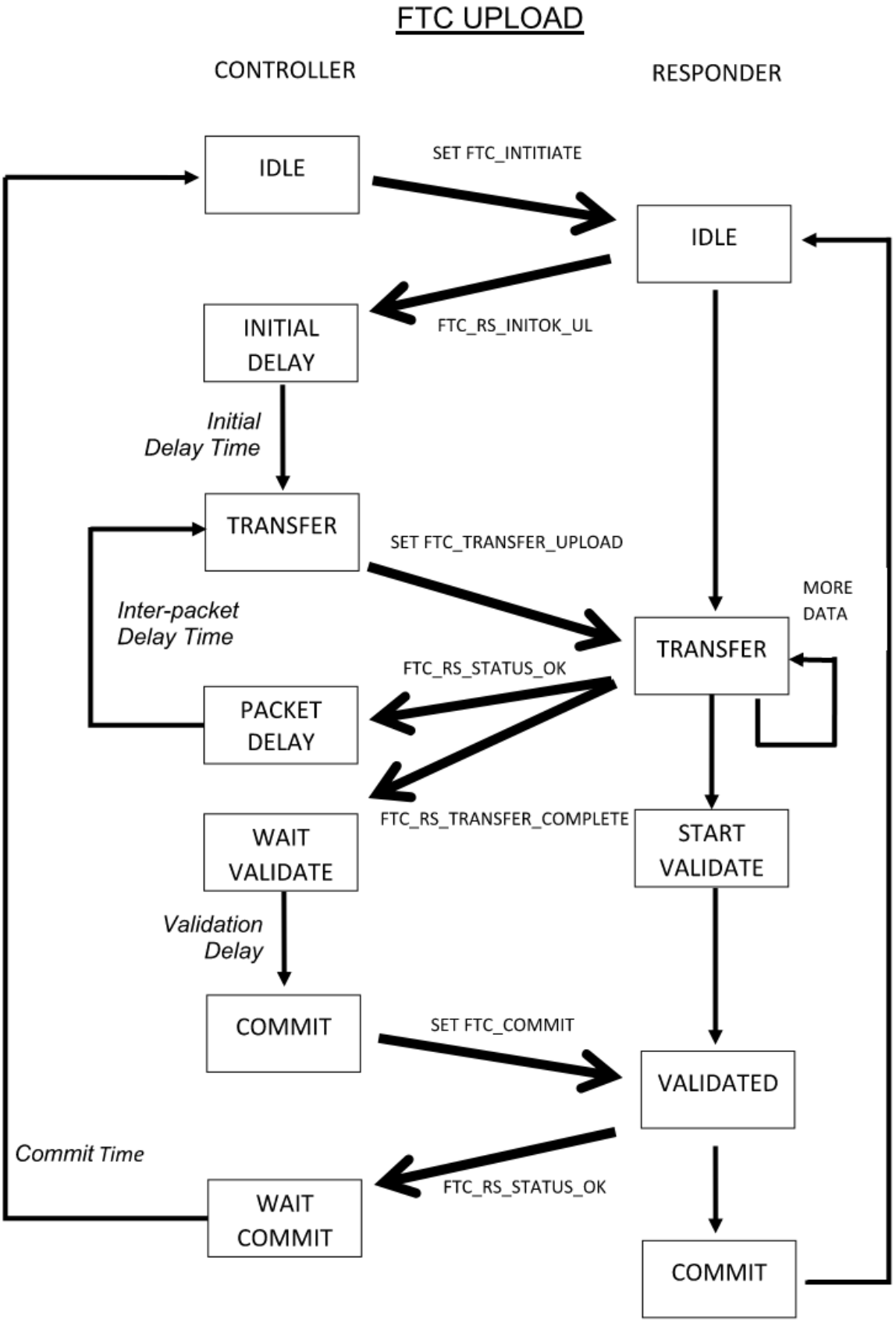
Responders shall treat invalid or out of range values in the FTC_INITIATE, FTC_TRANSFER_UPLOAD, FTC_TRANSFER_DOWNLOAD, FTC_COMMIT and FTC_CANCEL Parameter Messages differently to other [RDM] responses.

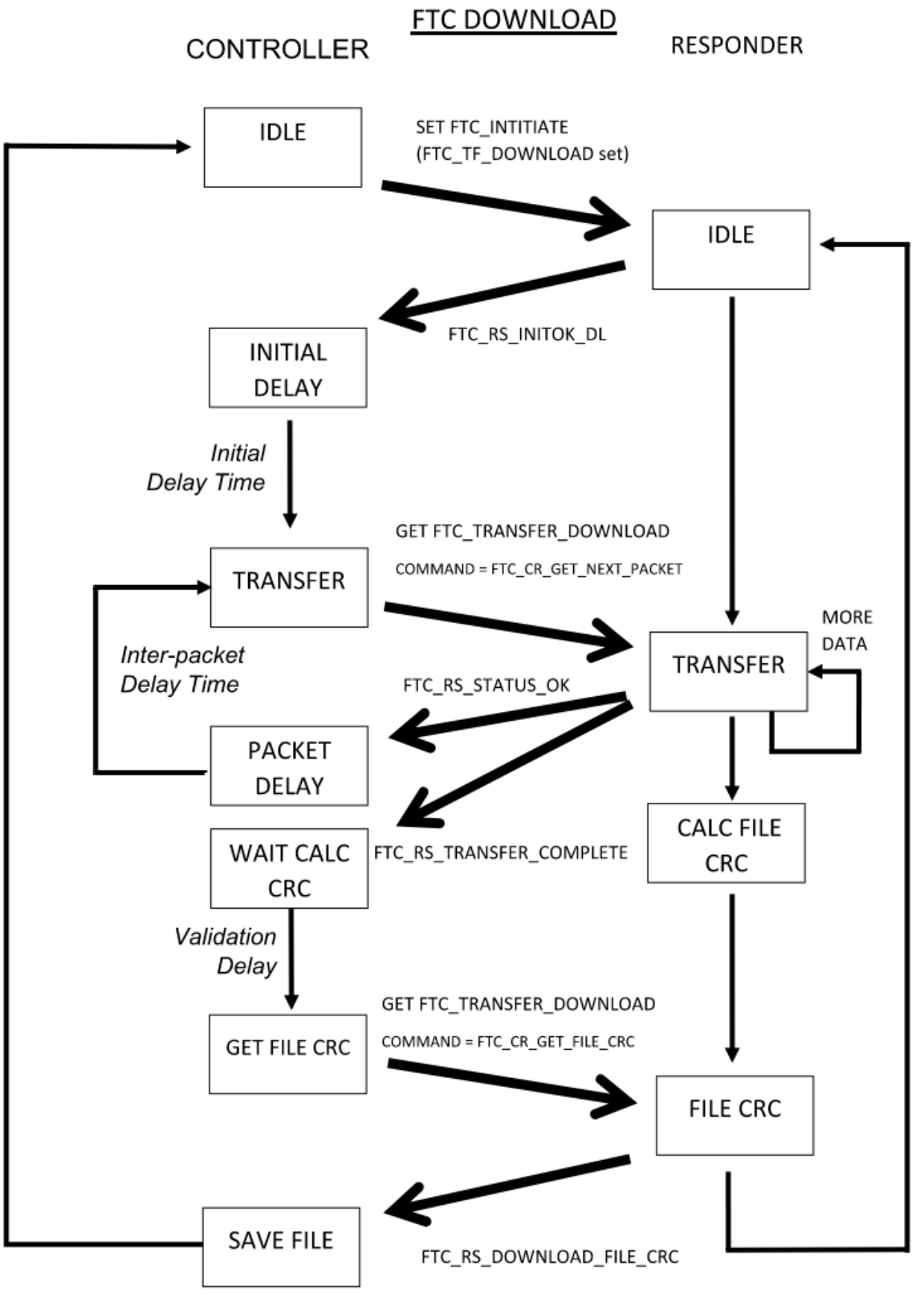
Responders shall not use the [RDM] NACK: DATA_OUT_OF_RANGE response when replying to these FTC Parameter Messages.

Responders detecting invalid or out of range parameter data values shall ACK the request and report the error using the ResponseStatus and ResponseData fields as described in Section 12.1 of this standard.

9. Operational Overview

Controller and Responder support for the Parameter Message is required as noted in Table A-2.




FTC DOWNLOAD
CONTROLLER
RESPONDER
IDLE
SET FTC_INTITIATE
(FTC_TF_DOWNLOAD set)
IDLE
FTC_RS_INITOK_DL
INITIAL DELAY
Initial Delay Time
GET FTC_TRANSFER_DOWNLOAD
COMMAND = FTC_CR_GET_NEXT_PACKET
TRANSFER
MORE DATA
TRANSFER
Inter-packet Delay Time
FTC_RS_STATUS_OK
PACKET DELAY
CALC FILE CRC
WAIT CALC CRC
FTC_RS_TRANSFER_COMPLETE
Validation Delay
GET FTC_TRANSFER_DOWNLOAD
COMMAND = FTC_CR_GET_FILE_CRC
GET FILE CRC
FILE CRC
SAVE FILE
FTC_RS_DOWNLOAD_FILE_CRC

9.1 Initiate Transfer

To obtain details about a possible file transfer, the Controller may issue a GET:FTC_INITIATE Parameter Message in accordance with Section 13.1.1 of this standard. This message allows collection of required transfer configuration information without preparing a Responder for a new data transfer.

To initiate a file transfer in either direction (Upload or Download), the Controller shall issue a SET:FTC_INITIATE Parameter Message in accordance with Section 13.1.3 of this standard.

9.2 Transfer File Data

Transfer of file data shall use the FTC_TRANSFER_UPLOAD Parameter Message in accordance with Section 13.2 or the FTC_TRANSFER_DOWNLOAD Parameter Message in accordance with Section 13.6 of this standard.

Files shall be transferred using an [RDM] packet size in accordance with characteristics obtained from the Responder.

The amount of file data that can be transferred in each [RDM] packet shall be determined from the Responder during the Initiate stage.

Failure or interruption of the data transfer process, or corruption of data shall not in any way affect the subsequent operational integrity of the Responder.

Controllers shall treat receipt of a ResponseStatus FTC_RS_TRANSFER_COMPLETE in a SET:FTC_TRANSFER_UPLOAD or GET:FTC_TRANSFER_DOWNLOAD response as confirmation that the transfer of File Data is complete.

9.3 Validation

Responders, having issued a ResponseStatus of FTC_RS_TRANSFER_COMPLETE to a SET:FTC_TRANSFER_UPLOAD, shall proceed with internal validation of the transferred File Data.

Validation determines that the data is suitable for the target Responder. Validation may include checking of the FileCRC or other means as determined by the Responder manufacturer.

Controllers shall use any declared Validation Delay Time before continuing to the Commit stage.

9.4 Commit File Data

The GET:FTC_COMMIT Parameter Message may be used to verify that the Responder has completed Validation and is ready to progress to the Commit stage. This Parameter Message should only be issued once the Transfer has been completed

The SET:FTC_COMMIT Parameter Message shall be used, once the Transfer has been completed and the Validation Delay Time has expired, to save the file as required.

When the file being uploaded requires a Responder to reboot, the SET:FTC_COMMIT is the only Parameter Message that instructs the Responder to reboot to use the transferred file. Firmware files are an example of where this would be expected.

The Responder response to a GET/SET:FTC_COMMIT returns a Commit Time and the anticipated Responder [RDM] UID after any reboot has occurred.

Controllers shall interpret the Commit Time as the time required by the Responder to complete any file save, firmware reprogramming and reboot time, during which it is reasonable to expect that the Responder is off-line and unable to respond to any [RDM] or FTC Parameter Messages.

Controllers shall not expect to resume communication with a Responder until after the Commit Time has expired.

Controllers shall take into account the returned ExpectedUID as part of the GET/SET:FTC_COMMIT response.

The FTC_COMMIT Parameter Message is described in Section 13.3 of this standard.

Controllers may set the FTC_TF_TESTMODE bit to allow transfer and validation of the file as normal, but without the Responder completing any file save or firmware reprogramming. Responders reply to the GET/SET FTC_COMMIT in the same way as normal.

9.5 Cancellation of Transfer

The FTC_CANCEL Parameter Message shall be used when the Controller wishes to abandon any transfer in accordance with Section 13.4 of this standard.

FTC_CANCEL may be sent after sending SET:FTC_COMMIT but may be ignored by the Responder.

A Controller should not rely on any data being held by the Responder after any cancellation of the transfer.

9.6 Use of Download Keys

Download Keys provide a means of restricting access to downloadable data and are defined by the Responder manufacturer.

Controllers can determine from the Responder Capabilities a need for a Download key for the requested download.

Controllers that do not support Download Keys should notify the user if the Responder has indicated in the Responder Capabilities a need for a Download key for the requested download. The method of such notification is beyond the scope of this standard.

10. Cyclic redundancy Check (CRC)

10.1 CRC Description

A cyclic redundancy check (CRC) is an error detecting code used to confirm the integrity of digital data by detecting accidental changes. The block of digital data is provided a small check value based on the remainder of a polynomial division of the block contents. The calculation is repeated during the check process (such as after receiving a transmitted block of data). If the values do not match then Devices can assume the data integrity has been compromised and take action accordingly. The CRC used in this standard is a 16-bit CRC, specifically CRC-16-ANSI (also known as CRC-16-IBM or Modbus CRC16). Additional information on CRC use and generation can be found in Appendix D or [Modbus] Appendix C.

10.2 Use of CRC

Where mention is made of FileCRC or PacketCRC in this standard, the following implementation shall be followed.

The CRC shall be the 16-bit remainder after division modulo 2 of the message polynomial by the generator polynomial x16+x15+x2+1, calculated according to the algorithm described in Appendix D.

The calculated CRC shall be transmitted in Big-Endian Order in accordance with [RDM] Section 6.1.

10.3 FileCRC

Controllers shall provide a FileCRC as part of the SET:FTC_INITIATE Parameter Message for uploads.

Responders may provide FileCRC as part of the GET:FTC_TRANSFER_DOWNLOAD Parameter Message for downloads. When provided, Controllers shall calculate the CRC on the whole file data received and verify it matches the returned FileCRC.

FileCRC is computed for the entire file contents.

Responders that declare capability for processing FileCRC shall process FileCRC before any other validation of the file content.

10.4 PacketCRC

Controllers shall provide a PacketCRC as part of the FTC_INITIATE Parameter Message.

Controllers shall provide a PacketCRC as part of the FTC_TRANSFER_UPLOAD Parameter Message for uploads.

Responders may provide a PacketCRC as part of the FTC_TRANSFER_DOWNLOAD Parameter Message for downloads.

The PacketCRC shall be calculated from the first byte of the Parameter Data for a length of Parameter Data Length minus 2.

For example, if the PDL is 16, CRC will be calculated over the first 14 bytes of the Parameter Data.

This will include the FTC header and the File Data, but not the PacketCRC field itself.

11. SessionID, FileID, and FTC Version

11.1 Use of SessionID

The SessionID is a 8-Bit field used to uniquely identify a file transfer session between a Controller and a Responder. It is created by the Controller and offered to the Responder as part of the Initiate stage. It is then used in the Transfer, Commit, and Cancel stages.

The SessionID shall be in the range 0x01-0xFE [as defined in Table A-1] and Controllers maintain the SessionID until it is committed or abandoned. Steps should be taken to randomize the SessionID after Controller startup to reduce repeated use of the same SessionID.

SessionIDs allow that:

- A Responder that has been disconnected during a file transfer, and subsequently reconnected during a different transfer has a means of rejecting the data as the new transfer should have a different SessionID.
- A Responder that has been relocated onto another physical data link that is in the process of transferring file data can decide to resume or reject the transfer.

The SessionID FTC_DEF_SESSIONID_ALL is reserved for use when a Controller wishes to cancel all current transfers.

Responders shall verify that the SessionID for the Transfer and Commit stages match the SessionID supplied in the SET:FTC_INITIATE Parameter Message.

11.2 Use of FileID

The FileID field is a unique defined value that identifies the file to be transferred.

Responders may assign values to represent the different files that are supported. They do not need to be consecutive and are not required to be the same for different Responder models. The valid range of FileIDs is defined in Table A-1.

The FileID values of FTC_DEF_NO_FILEID_OFFERED and FTC_DEF_MULTIPLE_FILEID are reserved and have specific meaning.

The value FTC_DEF_NO_FILEID_OFFERED may be sent from the Controller to the Responder in the GET:FTC_INITIATE Parameter Message to determine if the Responder supports one or more FileIDs.

If the Responder only supports a single file, the Responder shall return a value in the range as listed in Table A-1 and it is this FileID that should be sent by the Controller in the SET:FTC_INITIATE Parameter Message.

The value FTC_DEF_NO_FILEID_OFFERED may also be used with SET: FTC_INITIATE in place of using GET:FTC_INITIATE in accordance with Sections 13.1.2 and 13.1.4.

Controllers should not assume that any given FileID represents a given file type. The GET:FTC_FILELIST Capabilities Bit Field should be used to determine the specific file type details.

11.3 Use of FTCVersion

FTCVersion is a 16-bit field used to indicate the major and minor version of the FTC Standard being used by the Controller or Responder.

Responders shall reply with FTC_RS_FTCVERSION_ERROR if a GET:FTC_INTITIATE or SET:FTC_INITIATE is received with an FTCVersion that is unsupported by the Responder.

Valid versions are:

Version (Major.Minor)	Version Description
0.01	Initial draft version, 2024-02-15
0.02	Draft version, 2024-04-11
0.03	Draft version, 2024-06-20
0.04	Draft version, 2024-10-26 [post Plugfest]
0.05	Draft version, 2025-02-05
0.xx	Future draft versions
xx.xx	Future ratified versions

Table 11-1: FTCVersion

This version of the FTC standard shall be referenced as FTCVersion 0.05.

Amend this table to just show Version 1.00 at time of ratification & publication

12. Replies from Responders

All replies from a Responder to the FTC_INITIATE, FTC_TRANSFER_UPLOAD, FTC_TRANSFER_DOWNLOAD, FTC_COMMIT and FTC_CANCEL Parameter Messages include the ResponseStatus and ResponseData fields.

12.1 ResponseStatus and ResponseData Field Use

ResponseStatus codes are defined in Table A-3, and Table A-4 details the applicable Parameter Messages.

12.1.1 Good response – OK to continue

The ResponseStatus of FTC_RS_STATUS_OK is used to indicate successful execution of a request except as described in Sections 12.1.2 and 12.1.3.

The ResponseData field shall be set to 0x00000000.

12.1.2 Good response – Initiate Complete

The ResponseStatus of FTC_RS_INITOK_UL or FTC_RS_INITOK_DL is used to indicate successful execution of an Initiate request based on the requested transfer direction.

The ResponseData field shall be set to 0x00000000.

12.1.3 Good response – Transfer Complete

The ResponseStatus of FTC_RS_TRANSFER_COMPLETE is used to indicate successful completion of a file transfer, either Upload or Download.

The ResponseData field shall be set to 0x00000000.

12.1.4 Good response – More time required

The ResponseStatus FTC_RS_STATUS_IN_PROGRESS informs the Controller that the Responder requires further time to action the request.

The ResponseData field shall be used to return the delay time (in ms).

The valid range is 0x00000000-0x00400000. (Max 4,194,304 ms or approx. 1.2hrs)

The delay time returned in this field becomes the Inter-Packet Delay Time for the next request.

In the absence of any delay time reported via the FTC_RS_STATUS_IN_PROGRESS response, a Controller shall use the default Inter-Packet Delay timing provided during the Initiate stage.

> *Although this has functionality similar to the use of ACK_TIMER as defined in [RDM], the ACK_TIMER granularity of 100ms is considered too coarse for timely upload of data that may require less than 10ms for the data to be saved by a Responder in its non-volatile storage.*

The use of FTC_RS_STATUS_IN_PROGRESS is not equivalent to the ACK_TIMER in so far as no use of QUEUED_MESSAGES is implied or required.

> *Controllers in receipt of a value out of range are expected to cancel the process or default to the maximum valid value.*

12.1.5 Good response – Switch to Bootloader

The ResponseStatus FTC_RS_SWITCH_TO_BOOTLOADER informs the Controller that the Responder requires time to switch to a Bootloader or alternate application in order to process further FTC Parameter Messages.

The ResponseData field shall be used to return the delay time (in ms) that the switch over requires.

The valid range is 0x00000000-0x00400000. (Max 4,194,304 ms or approx. 1.2hrs).

Controllers shall ignore all other data from the initial SET:FTC_INITIATE response.

Controllers shall send a new SET:FTC_INITIATE after the delay in order to initiate any transfer. The Controller may reuse the previous SessionID but is not required to do so.

Controllers in receipt of a value out of range are expected to cancel the process.

12.1.6 Good response – Download FileCRC available

The ResponseStatus FTC_RS_DOWNLOAD_FILE_CRC informs the Controller that the Responder has calculated a FileCRC for the download.

The ResponseData field shall contain the FileCRC in the lower 16bits. The upper 16 bits shall be set to 0x0000.

12.1.7 Bad response – Modal Error

The ResponseStatus FTC_RS_MODAL_ERROR informs the Controller that the Responder was not expecting the Parameter Message at this stage of the FTC process.

The ResponseData field may contain a Manufacturer Specific error code in the lower 16bits for the purpose of diagnosis or debugging. The upper 16bits shall be set to 0x0000.

Interpretation of Manufacturer Specific codes by Controllers is beyond the scope of this standard.

Responders shall use FTC_RS_MODAL_ERROR and not FTC_RS_SESSIONID_MISMATCH when in receipt of Parameter Messages requiring verification of SessionID (FTC_TRANSFER_UPLOAD, FTC_TRANSFER_DOWNLOAD, FTC_COMMIT, FTC_CANCEL) prior to a successful Initiate.

12.1.8 Bad response – SessionID Mismatch

The ResponseStatus FTC_RS_SESSIONID_MISMATCH informs the Controller that the Responder was not expecting the SessionID or that it does not match the SessionID offered during Initiate.

The ResponseData field shall contain the received SessionID value in the upper 16 bits and the expected SessionID value in the lower 16 bits as detailed here.

ResponseData Field			
Upper 16bits		Lower 16bits	
0x00	Received SessionID	0x00	Expected SessionID

The expected SessionID is that offered in the SET:FTC_INITATE, except for the GET:FTC_INITIATE Parameter Message where the expected SessionID is FTC_DEF_NO_SESSIONID_OFFERED.

12.1.9 Bad response – PacketCRC Error

The ResponseStatus FTC_RS_PACKET_CRC_ERROR informs the Controller that the Responder calculation of the PacketCRC does not match the received PacketCRC.

The ResponseData field shall contain the received PacketCRC value in the upper 16 bits and the calculated PacketCRC value in the lower 16 bits.

12.1.10 Bad response – FileCRC Error

The ResponseStatus FTC_RS_FILE_CRC_ERROR informs the Controller that the Responder calculation of the FileCRC does not match the received FileCRC.

The ResponseData field shall contain the received FileCRC value in the upper 16 bits and the calculated FileCRC value in the lower 16 bits.

Responders that declare capability for processing FileCRC shall process FileCRC before any other validation of the file content.

12.1.11 Bad response – Validation Error

The ResponseStatus FTC_RS_VALIDATION_ERROR informs the Controller that the Responder's File Data validation has failed and cannot continue to the Commit stage.

The ResponseData field may contain a Manufacturer Specific error code in the lower 16bits for the purpose of diagnosis or debugging. The upper 16bits shall be set to 0x0000.

Interpretation of Manufacturer Specific codes by Controllers is beyond the scope of this standard.

12.1.12 Bad response – Unsupported FileID

The ResponseStatus FTC_RS_UNSUPPORTED_FILEID informs the Controller that the Responder does not support the offered FileID.

The ResponseData field shall contain the Supported FileID in the lower 16bits and the Offered FileID in the upper 16 bits as detailed here.

ResponseData Field			
Upper 16bits		Lower 16bits	
0x00	Offered FileID	0x00	Supported FileID

The Supported FileID value shall be as defined in Table A-1 to indicate the single file supported or set to FTC_DEF_MULTIPLE_FILEID to indicate that multiple files are supported.

12.1.13 Bad response – Invalid Direction

The ResponseStatus FTC_RS_INVALID_DIRECTION informs the Controller that the Responder does not support the requested transfer direction for the offered FileID.

The ResponseData field shall be set to 0x00000000.

12.1.14 Bad response – File Not Compatible

The ResponseStatus FTC_RS_FILE_NOT_COMPATIBLE informs the Controller that the Responder has rejected the File Data as being incompatible with the Responder.

The ResponseData field may contain a Manufacturer Specific error code in the lower 16bits for the purpose of diagnosis or debugging. The upper 16bits shall be set to 0x0000.

Interpretation of Manufacturer Specific codes by Controllers is beyond the scope of this standard.

12.1.15 Bad response – File Size Error

The ResponseStatus FTC_RS_FILESIZE_ERROR informs the Controller that the Responder is rejecting the offered File Size, or that the Transfer data is beyond the declared File Size.

The ResponseData field shall be set to the expected File Size.

12.1.16 Bad response – File Not Available

The ResponseStatus FTC_RS_FILE_NOT_AVAILABLE informs the Controller that the Responder File Data is not currently available.

The ResponseData field may contain a Manufacturer Specific error code in the lower 16bits for the purpose of diagnosis or debugging. The upper 16bits shall be set to 0x0000.

Interpretation of Manufacturer Specific codes by Controllers is beyond the scope of this standard.

12.1.17 Bad response – Offset Error

The ResponseStatus FTC_RS_OFFSET_ERROR informs the Controller that the Responder was not expecting the File Offset as received in the last SET:FTC_TRANSFER_UPLOAD or GET:FTC_TRANSFER_DOWNLOAD Parameter Message, or that it was out of range.

The ResponseData field shall be set to the expected File Offset.

12.1.18 Bad response – Other Conditions

Responders shall return FTC_RS_FTCVERSION_ERROR if they do not support the version of the FTC process offered by the Controller in a GET:FTC_INITIATE or SET:FTC_INITIATE Parameter Message.

The ResponseData field upper 16bits shall be set to the Major Version and the lower 16bits shall be set to the Minor Version supported by the Responder.

Responders may return FTC_RS_E137_LOCKACTIVE or FTC_RS_OTHER_LOCKACTIVE if there is a currently active lock mechanism preventing access to the details associated with the FileID.

Responders may return FTC_RS_WRITE_PROTECT if a memory protection scheme is preventing acceptance of data. For example, an SD memory card may be WRITE_PROECTED.

Responders may return FTC_RS_UNRESOLVED_ERROR to report any other error condition.

The ResponseData field may contain a Manufacturer Specific error code in the lower 16bits for the purpose of diagnosis or debugging. The upper 16bits shall be set to 0x0000.

12.1.19 Manufacturer Specific Conditions

The use of Manufacturer Specific ResponseStatus codes is permitted, but not encouraged, as interpretation of any such codes by Controllers is beyond the scope of this standard.

13. Parameter Messages Use and Definition

Responders implementing Parameter Messages shown as GET only in Table A-2 shall treat any SET_COMMAND as an error and report NRC_UNSUPPORTED_COMMAND_CLASS in accordance with [RDM].

Responders implementing Parameter Messages shown as SET only in Table A-2 shall treat any GET_COMMAND as an error and report NRC_UNSUPPORTED_COMMAND_CLASS in accordance with [RDM].

13.1 Initiate Transfer (FTC_INITIATE)

The SET:FTC_INITIATE Parameter Message provides information about the direction of the transfer, the size of file being offered, and the response returns details to be used by the Controller to manage the process.

A Controller shall issue a SET:FTC_INITIATE Parameter Message to inform the Responder that it is about to commence file transfer using one or more FTC_TRANSFER_UPLOAD or FTC_TRANSFER_DOWNLOAD Parameter Messages

Controllers may use the GET:FTC_FILELIST Parameter Message to determine what files and file types are supported. Files may be for upload, download, or both as determined from the Capabilities Bit Field returned with the GET:FTC_FILELIST Parameter Message.

Controllers may use the GET:FTC_INITIATE Parameter Message to ascertain the capabilities of a Responder prior to initiating a transfer using SET:FTC_INITIATE Parameter Message. The GET:FTC_INITIATE does not tell the responder anything about a pending data transfer.

13.1.1 Controller: Initiate Transfer (GET:FTC_INITIATE)

Controller: (GET)

(Port ID) 0x01-0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) FTC_INITIATE	(PDL) 0x06

(PD)

SessionID (8-bit)
FileID (8-bit)
FTCVersion (16-bit)
TransferFlags (16-bit)

Data Description:

SessionID (8-bit):
Controllers shall set this to FTC_DEF_NO_SESSIONID_OFFERED as defined in Table A-1 and Section 11.1 of this standard.

FileID (8-bit):
Controllers shall set this to FTC_DEF_NO_FILEID_OFFERED or a known valid FileID as defined in Table A-1 and Section 11.2 of this standard.

FTCVersion (16-bit):
Controllers shall set this field to indicate the major and minor version of the FTC Standard being used by the Controller in accordance with Section 11.3 of this standard.

TransferFlags (16-bit):
Controllers shall set this field to indicate transfer information as defined in Table A-5. This field determines the desired transfer direction, Upload or Download.

13.1.2 Response to GET:FTC_INITIATE

Responder: (GET Response) – – FTC_INITIATE

(Response Type) ACK	(Message Count) 0x00	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) FTC_INITIATE	(PDL) 0x22

(PD)

ResponseStatus (8-bit)
ResponseData (32-bit)
SessionID (8-bit)
FileID (8-bit)
FTCVersion (16-bit)
File Size (32-bit)
Capabilities Bit Field (32-bit)
DataOffset (32-bit)
TransferBlock Size (8-bit)
Initial Delay Time (32-bit)
Inter-Packet Delay Time (16-bit)
Accumulated Byte Count (32-bit)
Accumulated Byte Delay Time (16-bit)
Validation Delay Time (16-bit)
Max Inter-Packet Delay Time (16-bit)

Data Description:

ResponseStatus (8-bit):
Responders shall return a ResponseStatus of FTC_RS_STATUS_OK when the SessionID offered in the GET_COMMAND is FTC_DEF_NO_SESSIONID_OFFERED, and the FileID offered is either FTC_DEF_NO_FILEID_OFFERED or is a supported FileID.

Responders shall return a ResponseStatus of FTC_RS_SESSIONID_MISMATCH if the SessionID offered is not FTC_DEF_NO_SESSIONID_OFFERED.

Responders shall return a ResponseStatus of FTC_RS_UNSUPPORTED_FILEID if the SessionID offered is FTC_DEF_NO_SESSIONID_OFFERED, and the Responder does not support the FileID offered.

Responders may return FTC_RS_E137_LOCKACTIVE, FTC_RS_OTHER_LOCKACTIVE or FTC_RS_WRITE_PROTECT if there is a currently active lock mechanism or write protect scheme preventing access to details associated with FileIDs.

Other Status conditions may be reported in accordance with Section 12.1 and Table A-4 of this standard.

ResponseData (32-bit):
Responders shall set this field in accordance with Section 12.1 of this standard.

SessionID (8-bit):
Responders not in a current FTC transfer process shall set this field to FTC_DEF_NO_SESSIONID_OFFERED.

Responders that have begun a FTC Transfer process shall set this field to the SessionID offered by the most recent successful SET:FTC_INITIATE.

FileID (8-bit):
Responders not in a current FTC Transfer process shall set the FileID value to indicate the file(s) supported.

Responders supporting a single FileID shall set a valid value as defined in Table A-1.

Controllers shall use this FileID in the SET:FTC_INITIATE Parameter Message.

Responders supporting multiple FileIDs shall set this field to FTC_DEF_MULTIPLE_FILEID.

If FileID is returned as FTC_DEF_MULTIPLE_FILEID the Controller should send a GET:FTC_FILELIST to determine the appropriate files, file types, and capabilities (by FileID). A Controller should have some means of presenting the user with the list of supported Files and a means of user selection of the required file to offer the Responder in order to complete the Initiate stage.

Responders that have begun a FTC Transfer process shall report the FileID offered by the most recent successful SET:FTC_INITIATE.

FTCVersion (16-bit):
This field is used to indicate the major and minor version of the FTC Standard being used by the Responder.

Responders shall set this field in accordance with Section 11.3 of this standard.

File Size (32-bit):
Responders should set this field for an Upload to the total File Size in bytes it is capable of receiving. If the file size is not known the Responder shall respond with 0x00000000.

Responders shall set this field for a Download to the total File Size in bytes. If the file size is not known the Responder shall respond with 0x00000000.

Capabilities Bit Field (32-bit):
Responders shall set this field in accordance with Section 8.6.7 and Table A-7 of this standard.

DataOffset (32-bit):
Responders shall set this field to return the current file offset.

For an Upload: Responders shall return the next expected file offset. This should equate to the total number of bytes of File Data successfully received and processed by the Responder for the last FTC_TRANSFER_UPLOAD.

Where no data has yet been received by a Responder this field shall be set to 0x00000000.

For a Download: Responders shall return the data offset for the last FTC_TRANSFER_DOWNLOAD sent by the Responder. Where no data has yet been provided by a Responder this field shall be set to 0x00000000.

Controllers may use this information to recover or continue a transfer.

TransferBlock Size (8-bit):
Responders shall set this field in accordance with Section 8.6.1 of this standard.

Controllers shall use this field in accordance with Section 8.6.1 of this standard.

Initial Delay Time (32-bit):
Responders shall set this field in accordance with Section 8.6.2 of this standard.

Controllers shall use this field in in accordance with Section 8.6.2 of this standard.

Inter-Packet Delay Time (16-bit):
Responders shall set this field in accordance with Section 8.6.3 of this standard.
Controllers shall use this field in in accordance with Section 8.6.3 of this standard.

Accumulated Byte Count (32-bit):
For an Upload: Responders shall set this field in accordance with Section 8.6.4 of this standard.

For a Download: Responders shall set this field to 0x0000.

Controllers shall use this field in in accordance with Section 8.6.4 of this standard.

Accumulated Byte Delay Time (16-bit):
For an Upload: Responders shall set this field in accordance with Section 8.6.5 of this standard.

For a Download: Responders shall set this field to 0x0000.

Controllers shall use this field in accordance with Section 8.6.5 of this standard.

Validation Delay Time (16-bit):
Responders shall set this field in accordance with Section 8.6.6 of this standard.

Controllers shall use this field in accordance with Section 8.6.6 of this standard.

Max Inter-Packet Delay Time (16-bit):
For an Upload: Responders shall set this field in accordance with Section 8.6.8 of this standard.

For a Download: Responders shall set this field to 0x0000.

Controllers shall use this field in accordance with Section 8.6.8 of this standard.

13.1.3 Controller: Initiate Transfer (SET:FTC_INITIATE)

The SET:FTC_INITIATE Parameter Message is used to inform the Responder of a file transfer using one or more FTC_TRANSFER_UPLOAD or FTC_TRANSFER_DOWNLOAD Parameter Messages and specifies the direction of the transfer and for Upload, information about the size of file being offered. The Responder returns details to be used by the Controller to manage the transfer process.

A Controller shall issue a SET:FTC_INITIATE Parameter Message to prepare a Responder for a file transfer and place it in the Initiate stage.

Controllers that have already ascertained a Responders FileID support using GET:FTC_INITIATE or GET_FTC_FILE_LIST should offer that single FileID to the Responder.

The FileID FTC_DEF_NO_FILEID_OFFERED may be used to determine if the Responder supports one or more FileIDs.

Controller: (SET) *Controller request to initiate a file transfer.*

(Port ID) 0x01-0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root)
(CC) SET_COMMAND	(PID) FTC_INITIATE	(PDL) 0x1E

(PD)

SessionID (8-bit)
FileID (8-bit)
FTCVersion (16-bit)
TransferFlags (16-bit)
File Size (32-bit)
First 16 bytes of File Data
FileCRC (16-bit)
PacketCRC (16-bit)

Data Description:

SessionID (8-bit):
Controllers shall create a SessionID in accordance with Section 11.1 and Table A-1.

Controllers shall use the same SessionID until the FTC process is complete, cancelled, or otherwise abandoned.

FileID (8-bit):
Controllers shall set this to FTC_DEF_NO_FILEID_OFFERED or a known FileID in accordance with Section 11.2 and Table A-1.

FTCVersion (16-bit):
This field is used to indicate the major and minor version of the FTC Standard being used by the Controller.

Controllers shall set this in accordance with Section 11.3 of this standard.

TransferFlags (16-bit):
Controllers shall set this field to indicate required transfer information in accordance with Table A-5, inclusive of any Manufacturer Specific flags.

Test Mode
For normal transfers, a Controller shall clear the FTC_TF_TESTMODE bit.

For an Upload, a Controller may set the FTC_TF_TESTMODE bit to instruct a Responder to receive the File in Test mode. Test mode allows transfer and validation without actually saving the file data or modifying any firmware.

Transfer Direction
For an Upload, a Controller shall clear the FTC_TF_DOWNLOAD bit.

For a Download, a Controller shall set the FTC_TF_DOWNLOAD bit.

File Size (32-bit):
For an Upload, a Controller shall set this field to the total File Size in bytes.

For a Download, a Controller shall set this field to 0x00000000.

File Data (16 bytes):
For an Upload, a Controller shall set this field to the first 16 bytes of the file.

In the event that the File Size is less than 16 bytes, Controllers shall set the remaining bytes as 0x00.

A Responder may use this data to validate that the file to be transferred is appropriate to the Responder.

The 16 byte limit in the Initiate stage is to satisfy the requirements of memory limited Responders. The overall size of the FTC_INITIATE Parameter Message (PDL plus RDM header) is designed to be less than 64bytes. This allows for Responders which can only support small RDM packets to use this standard.

Where manufactures require more than 16 bytes to validate the file, there is nothing in this standard that prevents manufacturers of Responders rejecting the data transfer after actual transfer has commenced.

The data does not need to be stored by the Responder at this point, and shall be sent again by the Controller as part of the first packet transfer using the FTC_TRANSFER_UPLOAD Parameter Message.

Responders that do not verify suitability of the data using this method are still free to reject data transfers at any time during the upload process.

For a Download, Controllers may use this field to pass a Download Key to the Responder.

If a Download Key is not required, or not available to the Controller, the Controller shall set all bytes in this field as 0x00. If the supplied Download Key is less than 16 bytes, the Controller shall set the remaining bytes as 0x00.

Responders shall indicate a requirement for a Download Key as part of their declared capabilities in Table A-7 for each supported FileID.

FileCRC (16-bit):
For an Upload, a Controller shall set this field to the FileCRC of the file being offered, in accordance with the method referenced in Section 10.3 of this standard.

For a Download, a Controller shall set this field to 0x0000.

PacketCRC (16-bit):
Controllers shall set this field to the PacketCRC as calculated in accordance with Section 10.4 of this standard.

The PacketCRC shall be calculated from the first byte of the Parameter Data for a length of Parameter Data Length minus 2.

13.1.4 Response to SET:FTC_INITIATE File Upload (TransferFlags FTC_TF_DOWNLOAD bit CLEAR)

Responder: (SET Response) – – FTC_INITIATE (file transfer UPLOAD)

(Response Type) ACK	(Message Count) 0x00	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_RESPONSE	(PID) FTC_INITIATE	(PDL) 0x22

(PD)

ResponseStatus (8-bit)
ResponseData (32-bit)
SessionID (8-bit)
FileID (8-bit)
FTCVersion (16-bit)
File Size (32-bit)
Capabilities Bit Field (32-bit)
DataOffset (32-bit)
TransferBlock Size (8-bit)
Initial Delay Time (32-bit)
Inter-Packet Delay Time (16-bit)
Accumulated Byte Count (32-bit)
Accumulated Byte Delay Time (16-bit)
Validation Delay Time (16-bit)
Max Inter-Packet Delay Time (16-bit)

Data Description:

ResponseStatus (8-bit):

For Responders supporting a single FileID for Upload:

If the offered FileID in the SET_COMMAND is FTC_DEF_NO_FILEID_OFFERED:

Responders supporting a single FileID shall return a ResponseStatus of FTC_RS_INITOK_UL and a FileID value in the range as listed in Table A-1.

The Controller is now free to use the returned data from the SET:FTC_INITIATE response and proceed with the Transfer stage using SET:FTC_TRANSFER_UPLOAD Parameter Messages.

If the offered FileID in the SET_COMMAND is within the allowed range as defined in Table A-1:

Responders supporting the offered FileID shall return a ResponseStatus of FTC_RS_INITOK_UL and the offered FileID.

The Controller is now free to use the returned data from the SET:FTC_INITIATE response and proceed with the Transfer stage using SET:FTC_TRANSFER_UPLOAD Parameter Message.

Responders that do not support the offered FileID shall return a ResponseStatus of FTC_RS_UNSUPPORTED_FILEID in accordance with Section 12.1.12 of this standard.

For Responders supporting a single FileID for Download Only:

Responders supporting a single FileID but only for Download shall return a ResponseStatus of FTC_RS_INVALID_DIRECTION to an Upload request (SET_COMMAND TransferFlags FTC_TF_DOWNLOAD bit CLEAR).

For Responders supporting multiple FileIDs:

If the offered FileID in the SET_COMMAND is FTC_DEF_NO_FILEID_OFFERED:

Responders supporting multiple FileID shall return a ResponseStatus of FTC_RS_STATUS_OK and a FileID value of FTC_DEF_MULTIPLE_FILEID.

The Controller cannot proceed with the Transfer stage until it has offered the Responder a single FileID in the SET_COMMAND.

If the offered FileID in the SET_COMMAND is within the allowed range as defined in Table A-1:

Responders supporting the offered FileID for Upload shall return a ResponseStatus of FTC_RS_INITOK_UL.

The Controller is now free to use the returned data from the SET:FTC_INITIATE response and proceed with the Transfer stage using SET:FTC_TRANSFER_UPLOAD.

Responders that do not support the offered FileID shall return a ResponseStatus of FTC_RS_UNSUPPORTED_FILEID in accordance with Section 12.1.12 of this standard.

Responders that support the offered FileID, but for Download Only shall return a ResponseStatus of FTC_RS_INVALID_DIRECTION in accordance with Section 12.1.13 of this standard.

Responders shall return FTC_RS_E137_LOCKACTIVE, FTC_RS_OTHER_LOCKACTIVE or FTC_RS_WRITE_PROTECT if there is a currently active lock mechanism or write protect scheme preventing access to the details associated with FileIDs.

Responders may return FTC_RS_FILE_NOT_COMPATIBLE if the Responder determines that the File Data is not suitable, based on the initial 16 bytes of File Data offered in the SET_COMMAND.

Responders may return FTC_RS_FILESIZE_ERROR if the Responder determines that the File Size offered is not suitable.

Other Status conditions may be reported in accordance with Section 12.1 and Table A-4 of this standard.

ResponseData (32-bit):
Responders shall set this field in accordance with Section 12.1 of this standard.

SessionID (8-bit):
Responders shall set this field to the SessionID offered in the SET_COMMAND.

FileID (8-bit):
Responders that support the FileID offered in the SET_COMMAND shall set this field to the offered FileID.

If the offered FileID is not supported or is FTC_DEF_NO_FILEID_OFFERED, Responders shall set the FileID to a value as defined in Table A-1 to indicate the file supported or set to FTC_DEF_MULTIPLE_FILEID to indicate that multiple files are supported.

Note: *If FileID is set to FTC_DEF_MULTIPLE_FILEID then the Controller should send a GET:FTC_FILELIST to determine the appropriate files, file types, and capabilities (by FileID).*

FTCVersion (16-bit):
This field is used to indicate the major and minor version of the FTC Standard being used by the Responder.

Responders shall set this in accordance with Section 11.3 of this standard.

File Size (32-bit):
Responders shall set this field to the File Size offered in the SET_COMMAND.

Capabilities Bit Field (32-bit):
Responders shall set this field in accordance with Section 8.6.7 and Table A-7 of this standard.

Controllers should utilize the declared capabilities to optimize the operation of the transfer process, or inform the user of any limitations.

DataOffset (32-bit):
Responders shall set this field to return 0x00000000 representing the start of the File.

TransferBlock Size (8-bit):
Responders shall set this field in accordance with Section 8.6.1 of this standard.

Controllers shall use this value in accordance with Section 8.6.1 of this standard.

Initial Delay Time (32-bit):
Responders shall set this field in accordance with Section 8.6.2 of this standard.

Controllers shall use this value in accordance with Section 8.6.2 of this standard.

Inter-Packet Delay Time (16-bit):
Responders shall set this field in accordance with Section 8.6.3 of this standard.

Controllers shall use this value in accordance with Section 8.6.3 of this standard.

Accumulated Byte Count (32-bit):
Responders shall set this field in accordance with Section 8.6.4 of this standard.

Controllers shall use this value in accordance with Section 8.6.4 of this standard.

Accumulated Byte Delay Time (16-bit):
Responders shall set this field in accordance with Section 8.6.5 of this standard.

Controllers shall use this value in accordance with Section 8.6.5 of this standard.

Validation Delay Time (16-bit):
Responders shall set this field in accordance with Section 8.6.6 of this standard.

Controllers shall use this value in accordance with Section 8.6.6 of this standard.

Max Inter-Packet Delay Time (16-bit):
Responders shall set this field in accordance with Section 8.6.8 of this standard.

Controllers shall use this field in accordance with Section 8.6.8 of this standard.

13.1.5 Response to SET:FTC_INITIATE File Download (TransferFlags FTC_TF_DOWNLOAD bit SET)

Responder: (SET Response) – FTC_INITIATE (file transfer DOWNLOAD)

(Response Type) ACK	(Message Count) 0x00	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_RESPONSE	(PID) FTC_INITIATE	(PDL) 0x22

(PD)

ResponseStatus (8-bit)
ResponseData (32-bit)
SessionID (8-bit)
FileID (8-bit)
FTCVersion (16-bit)
File Size (32-bit)
Capabilities Bit Field (32-bit)
DataOffset (32-bit)
TransferBlock Size (8-bit)
Initial Delay Time (32-bit)
Inter-Packet Delay Time (16-bit)
Accumulated Byte Count (32-bit)
Accumulated Byte Delay Time (16-bit)
Validation Delay Time (16-bit)
Max Inter-Packet Delay Time (16-bit)

Data Description:

ResponseStatus (8-bit):

For Responders supporting a single FileID for Download:

If the offered FileID in the SET_COMMAND is FTC_DEF_NO_FILEID_OFFERED:

Responders supporting a single FileID shall return a ResponseStatus of FTC_RS_INITOK_DL and a FileID value in the range as listed in Table A-1.

The Controller is now free to use the returned data from the SET:FTC_INITIATE response and proceed with the Transfer stage using GET:FTC_TRANSFER_DOWNLOAD Parameter Messages.

If the offered FileID in the SET_COMMAND is within the allowed range as defined in Table A-1:

Responders supporting the offered FileID shall return a ResponseStatus of FTC_RS_INITOK_DL and a FileID value in the range as listed in Table A-1.

The Controller is now free to use the returned data from the SET:FTC_INITIATE response and proceed with the Transfer stage using GET:FTC_TRANSFER_DOWNLOAD Parameter Messages.

Responders that do not support the offered FileID shall return a ResponseStatus of FTC_RS_UNSUPPORTED_FILEID in accordance with Section 12.1.12 of this standard.

For Responders supporting a single FileID for Upload Only:

Responders supporting a single FileID but only for Upload shall return a ResponseStatus of FTC_RS_INVALID_DIRECTION to a Download request (SET_COMMAND TransferFlags FTC_TF_DOWNLOAD bit SET).

For Responders supporting multiple FileIDs

If the offered FileID in the SET_COMMAND is FTC_DEF_NO_FILEID_OFFERED:

Responders supporting multiple FileID shall return a ResponseStatus of FTC_RS_STATUS_OK and a FileID value of FTC_DEF_MULTIPLE_FILEID.

The Controller cannot proceed with the Transfer stage until it has offered the Responder a single FileID in the SET_COMMAND.

If the offered FileID in the SET_COMMAND is within the allowed range as defined in Table A-1:

Responders supporting the offered FileID for Download shall return a ResponseStatus of FTC_RS_INITOK_DL.

The Controller is now free to use the returned data from the SET:FTC_INITIATE response and proceed with the Transfer stage using GET:FTC_TRANSFER_DOWNLOAD.

Responders that do not support the offered FileID shall return a ResponseStatus of FTC_RS_UNSUPPORTED_FILEID in accordance with Section 12.1.12 of this standard.

Responders that support the offered FileID, but for Upload Only shall return a ResponseStatus of FTC_RS_INVALID_DIRECTION in accordance with Section 12.1.13 of this standard.

Responders may return FTC_RS_E137_LOCKACTIVE or FTC_RS_OTHER_LOCKACTIVE if there is a currently active lock mechanism preventing access to the details associated with FileIDs.

Responders shall return FTC_RS_DOWNLOAD_KEY_REQUIRED if the Responder required a Download Key and no valid Key was supplied.

Other Status conditions may be reported in accordance with Section 12.1 and Table A-4 of this standard.

ResponseData (32-bit):
Responders shall set this field in accordance with Section 12.1 of this standard.

SessionID (8-bit):
Responders shall set this field to the SessionID offered in the SET_COMMAND.

FileID (8-bit):
The Responder shall set the FileID value as defined in Table A-1 to indicate the file supported or set to FTC_DEF_MULTIPLE_FILEID to indicate that multiple files are supported.

Responders that have begun a FTC Transfer process shall report the FileID offered by the most recent successful SET:FTC_INITIATE.

Note: *If FileID is set to FTC_DEF_MULTIPLE_FILEID then the Controller should send a GET:FTC_FILELIST to determine the appropriate files, file types, and capabilities (by FileID).*

FTCVersion (16-bit):
This field is used to indicate the major and minor version of the FTC Standard being used by the Responder.

Responders shall set this in accordance with Section 11.3 of this standard.

File Size (32-bit):
Responders shall set this field shall be set to total File Size in bytes, of the file to be downloaded. If the file size is not known the Responder shall respond with 0x00000000.

Capabilities Bit Field (32-bit):
Responders shall set this field in accordance with Section 8.6.7 and Table A-7 of this standard.

Controllers shall use the declared capabilities to optimize the operation of the transfer process, or inform the user of any limitations.

DataOffset (32-bit):
Responders shall set this field to return 0x00000000 representing the start of the File.

TransferBlock Size (8-bit):
Responders shall set this field in accordance with Section 8.6.1 of this standard.

Controllers shall use this field in accordance with Section 8.6.1 of this standard.

Initial Delay Time (32-bit):
Responders shall set this field in accordance with Section 8.6.2 of this standard.

Controllers shall use this field in in accordance with Section 8.6.2 of this standard.

Inter-Packet Delay Time (16-bit):
Responders shall set this field in accordance with Section 8.6.3 of this standard.

Controllers shall use this field in in accordance with Section 8.6.3 of this standard.

Accumulated Byte Count (32-bit):
For a Download: Responders shall set this field to 0x0000.

Accumulated Byte Delay Time (16-bit):
For a Download: Responders shall set this field to 0x0000.

Validation Delay Time (16-bit):
Responders shall set this field in accordance with Section 8.6.6 of this standard.

Controllers shall use this field in accordance with Section 8.6.6 of this standard.

Max Inter-Packet Delay Time (16-bit):
Responders shall set this field to 0x0000.

The use of [RDM] ALL_DEVICES_ID is not applicable to a GET:FTC_TRANSFER_DOWNLOAD.

13.2 Transfer Data Upload (FTC_TRANSFER_UPLOAD)

The FTC_TRANSFER_UPLOAD Parameter Message is used to transfer data to a Responder.

Responders should only expect a data transfer after receipt of the SET:FTC_INITIATE Parameter Message.

Responders shall reject a FTC_TRANSFER_UPLOAD Parameter Message using the ResponseStatus FTC_RS_MODAL_ERROR if it is received before a SET:FTC_INITIATE has been processed by the Responder.

Responders shall test the SessionID given in the GET or SET Parameter Message against the SessionID given in the SET:FTC_INITIATE Parameter Message and return a ResponseStatus of FTC_RS_SESSIONID_MISMATCH if they are different.

13.2.1 Controller: Transfer Data to a Responder (SET:FTC_TRANSFER_UPLOAD)

Controllers shall use the TransferBlock Size returned by the Responder during the Initiate stage to determine the number of data bytes to be sent in each packet.

Controllers shall use the Initial Delay Time value returned by the Responder during the Initiate stage to control the time between the SET:FTC_INITIATE Parameter Message and the first packet sent using the FTC_TRANSFER_UPLOAD Parameter Message.

Controllers shall use the Inter-Packet Delay Time value returned by the Responder during the Initiate stage to control the time between each packet sent. Controllers performing One-to-Many transfers shall use the Max Inter-Packet Delay Time instead of the Inter-Packet Delay Time.

Where a Responder has declared a non zero Accumulated Byte Count and non zero Accumulated Byte Delay Time as part of the Initiate stage, the Controller shall use these values in addition to the Inter-Packet Delay time after every N bytes where N = Accumulated Byte Count.

Controller: (SET)

(Port ID) 0x01-0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root)
(CC) SET_COMMAND	(PID) FTC_TRANSFER_UPLOAD	(PDL) 0x07- 0xE7

(PD)

SessionID (8-bit)
DataOffset (32-bit)
Data (0-224 Bytes)
PacketCRC (16-bit)

Data Description:

SessionID (8-bit):
The SessionID is created and issued by a Controller at the Initiate stage in accordance with Section 11 of this standard.

Controllers shall set this field to the SessionID generated at the time of the Initiate stage.

DataOffset (32-bit):
Controllers shall set this field to indicate the byte offset within the file for the data in this packet.

This shall be used by the Responder to store the data.

Controllers shall increment the DataOffset by the TransferBlock Size after a receipt of a ResponseStatus FTC_RS_STATUS_OK for the SET:FTC_TRANSFER_UPLOAD Parameter Message.

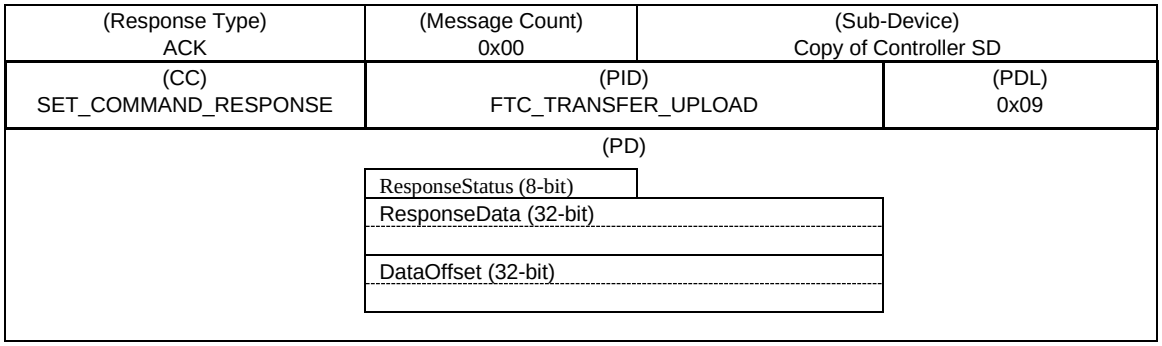
Data (0-224 Bytes):
This field contains the data being transferred from the Controller to the Responder. The length of the data shall be the TransferBlock Size for each packet sent, with the exception of the final packet where the length will be less than or equal to the TransferBlock Size.

PacketCRC (16-bit):
Controllers shall set this field to the PacketCRC as calculated in accordance with Section 10.4 of this standard.

The PacketCRC shall be calculated from the first byte of the Parameter Data for a length of Parameter Data Length minus 2.

13.2.2 Response to SET:FTC_TRANSFER_UPLOAD

Response: (SET:Response)



Data Description:

ResponseStatus (8-bit):
A ResponseStatus of FTC_RS_STATUS_OK indicates that this Transfer packet is correctly received and that the Responder is ready for more data.

A ResponseStatus of FTC_RS_TRANSFER_COMPLETE indicates that File Transfer is now complete.

Responders shall begin Validation, and calculation of the FileCRC, if supported, once the File Transfer is complete.

A ResponseCode of FTC_RS_STATUS_INPROGRESS indicates that processing of this Transfer packet is not yet complete. The Responder shall report the remaining required time (in ms) in the ResponseData field.

Controllers shall wait the specified time before continuing with the next FTC_TRANSFER_UPLOAD packet.

A Controller may issue a GET:FTC_TRANSFER_UPLOAD until the ResponseStatus returned is FTC_RS_STATUS_OK. Alternatively, the Controller may send the next SET:FTC_TRANSFER_UPLOAD after the delay.

A ResponseStatus of FTC_RS_PACKET_CRC_ERROR indicates that the calculated PacketCRC does not match for this packet. The Responder shall report in the ResponseData field the Received PacketCRC value in the upper 16 bits and the Calculated PacketCRC value in the lower 16 bits.

A Controller may elect to resend the packet or abandon the transfer. A Controller shall limit resends to a maximum of three attempts for any one packet.

In the event that the Controller decides to abandon the transfer (based on received ResponseStatus or resend count exceeded), it shall do so in accordance with Section 13.4 of this standard.

Other Status conditions may be reported in accordance with Section 12.1 and Table A-4 of this standard.

ResponseData (32-bit):
Responders shall set this field in accordance with Section 12.1 of this standard.

DataOffset (32-bit):
Responders shall return the next expected file offset. This should equate to the offset sent by the Controller in the SET_COMMAND plus the length of data stored by the Responder.

When returning the ResponseStatus of FTC_RS_TRANSFER_COMPLETE to indicate that File Transfer is now complete, the DataOffset shall be set to the total number of bytes of File Data successfully received and processed by the Responder. This should correspond to the File Size of the offered File.

If the Responder returns a ResponseStatus other than FTC_RS_STATUS_OK or FTC_RS_TRANSFER_COMPLETE the DataOffset shall not be incremented.

Controllers may verify the expected and actual offsets in order to determine successful processing of the packet by the Responder.

13.2.3 Controller: Getting Transfer Status FROM a Responder (GET:FTC_TRANSFER_UPLOAD)

Controllers may use this Parameter Message to check on the progress of a Transfer or to probe whether a delay resulting from a ResponseStatus of FTC_RS_STATUS_INPROGRESS has completed.

Use of the GET:FTC_TRANSFER_UPLOAD is not normally required in a One-to-One transaction but may be required for Controllers utilizing One-to-Many, to check on progress and determine if any retries are required.

Controller: (GET)

(Port ID) 0x01-0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) FTC_TRANSFER_UPLOAD	(PDL) 0x01
(PD) SessionID (8-bit)		

13.2.4 Response to GET:FTC_TRANSFER_UPLOAD

Response: (GET:Response)

(Response Type) ACK	(Message Count) 0x00	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) FTC_TRANSFER_UPLOAD	(PDL) 0x09
(PD) ResponseStatus (8-bit) ResponseData (32-bit) DataOffset (32-bit)		

Data Description:

ResponseStatus (8-bit):
A ResponseStatus of FTC_RS_STATUS_OK indicates that the last SET:FTC_TRANSFER_UPLOAD was correctly received and that the Responder is ready for more data.

A ResponseStatus of FTC_RS_STATUS_INPROGRESS indicates that the last SET:FTC_TRANSFER_UPLOAD is not yet complete. The Responder shall report the remaining required time (in ms) in the ResponseData field.

The Controller may issue a GET:FTC_TRANSFER_UPLOAD until the ResponseStatus returned is FTC_RS_STATUS_OK. At this point the Controller may send the next SET:FTC_TRANSFER_UPLOAD.

A ResponseStatus of FTC_RS_PACKET_CRC_ERROR indicates that the last SET:FTC_TRANSFER_UPLOAD is complete, but that the calculated PacketCRC did not match. The Responder shall report in the ResponseData field the Received PacketCRC value in the upper 16 bits and the Calculated PacketCRC value in the lower 16 bits.

A Controller may elect to resend the packet or abandon the transfer. A Controller shall limit resends to a maximum of three attempts for any one packet.

In the event that the Controller decides to abandon the transfer (based on received ResponseStatus or resend count exceeded), it shall do so in accordance with Section 13.4 of this standard.

Other Status conditions may be reported in accordance with Section 12.1 and Table A-4 of this standard.

A ResponseStatus of FTC_RS_TRANSFER_COMPLETE indicates that last SET:FTC_TRANSFER_UPLOAD was regarded by the Responder as the final packet in the file.

Other Status conditions may be reported in accordance with Section 12.1 and Table A-4 of this standard.

ResponseData (32-bit):
Responders shall set this field in accordance with Section 12.1 of this standard.

DataOffset (32-bit):
Responders shall return the next expected file offset. This should equate to the total number of bytes of File Data successfully received and processed by the Responder.

13.3 Commit Data (FTC_COMMIT)

A SET:FTC_COMMIT shall be used by a Controller to instruct a Responder to commit the data. The Responder is responsible for ensuring that the data has been validated and is appropriate for the device.

Controllers shall only issue this command following the successful receipt of FTC_RS_TRANSFER_COMPLETE in response to the FTC_TRANSFER_UPLOAD of the final Transfer packet.

Responders shall ensure that a response to the SET:FTC_COMMIT is issued prior to commencement of any firmware reset that results in the Responder going off-line.

Controllers shall expect that, following receipt of a good response to the commit, the Responder will go off-line.

Responders may not necessarily re-appear with the same [RDM] UID. It may be necessary to run a discovery process before further communications with the Responder(s) are possible.

Responders inform a Controller of an expected change of UID via the ExpectedUID field.

Responders shall test the SessionID given in the GET or SET Parameter Message against the SessionID given in the SET:FTC_INITIATE Parameter Message and return a ResponseStatus of FTC_RS_SESSIONID_MISMATCH if they are different.

Responders shall check the FTC_TF_TESTMODE bit in the CommitFlags. If set, the Responder shall reply to the GET/SET FTC_COMMIT with the CalculatedFileCRC as normal, but not complete any file save or firmware reprogramming. Any Commit Time shall be reported as normal, but not executed. The Responder shall return to Idle immediately using the current UID.

13.3.1 Controller: GET Request to establish Commit Requirements (GET:FTC_COMMIT)

The GET:FTC_COMMIT Parameter Message may be used to verify that the Responder has completed Validation and is ready to progress to the Commit stage. This Parameter Message should only be issued once the Transfer has been completed.

Controller: (GET)

(Port ID) 0x01-0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root)

(CC) GET_COMMAND	(PID) FTC_COMMIT	(PDL) 0x03
(PD)		
SessionID (8-bit)		
CommitFlags (16-bit)		

Data Description:

SessionID (8-bit):
Controllers shall set this to the SessionID relating to the Transfer that they now wish to verify has completed the Validate stage.

Controllers shall NOT use the value FTC_DEF_SESSIONID_ALL with this Parameter Message.

CommitFlags (16-bit):
Controllers shall set the FTC_TF_TESTMODE and FTC_TF_DOWNLOAD bits to match those sent in the SET:FTC_INITIATE.

CommitFlags bits are defined in Table A-5.

13.3.2 Response to GET:FTC_COMMIT

Responder: (GET)

(Response Type) ACK	(Message Count) 0x00	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) FTC_COMMIT	(PDL) ~~0x0E~~ 0x0F
(PD)		
ResponseStatus (8-bit)		
ResponseData (32-bit)		
CommitFlags (16-bit)		
CalculatedFileCRC (16-bit)		
ExpectedUID (6 bytes)		

Data Description:

ResponseStatus (8-bit):
A ResponseStatus of FTC_RS_STATUS_OK indicates that Validation is complete.

A ResponseStatus of FTC_RS_STATUS_INPROGRESS indicates that Validation is not yet complete.

A ResponseStatus of FTC_RS_FILE_CRC_ERROR indicates that Validation is complete, but that the calculated FileCRC does not match that offered during the SET:FTC_INITIATE.

A ResponseStatus of FTC_RS_MODAL_ERROR shall be used if the CommitFlags do not match the TransferFlags received in the SET:FTC_INITIATE.

Other Status conditions may be reported in accordance with Section 12.1 and Table A-4 of this standard

ResponseData (32-bit):
Responders shall set this in accordance with Section 12.1 of this standard.

CommitFlags (16-bit):
Responders shall set this to value offered as the CommitFlags in the GET_COMMAND.

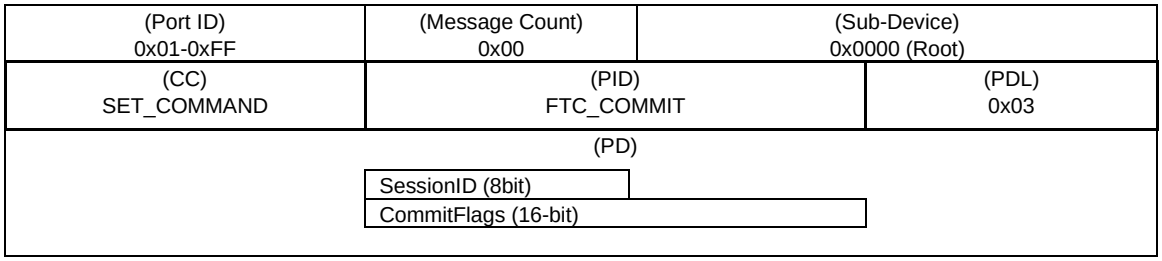
CalculatedFileCRC (16-bit):
The CRC that the Responder has calculated for the entire FTC Upload data received. Responders shall check this value against the FileCRC received from the Controller in the SET:FTC_INITIATE Parameter Message. A ResponseStatus of FTC_RS_STATUS_OK indicates that CalculatedFileCRC matches the FileCRC received from the Controller.

Responders unable to calculate the CalculatedFileCRC shall set this field to 0x0000.

ExpectedUID:
Responders shall set this to the expected [RDM] UID that the Responder will present after any firmware update and reboot, in accordance with Section 9.4 of this standard.

13.3.3 Controller: SET Request to Commit Data (SET:FTC_COMMIT)

Controller: (SET)



Data Description:

SessionID (8-bit):
Controllers shall set this to the SessionID relating to the Transfer that has been Validated and it now wishes to Commit.

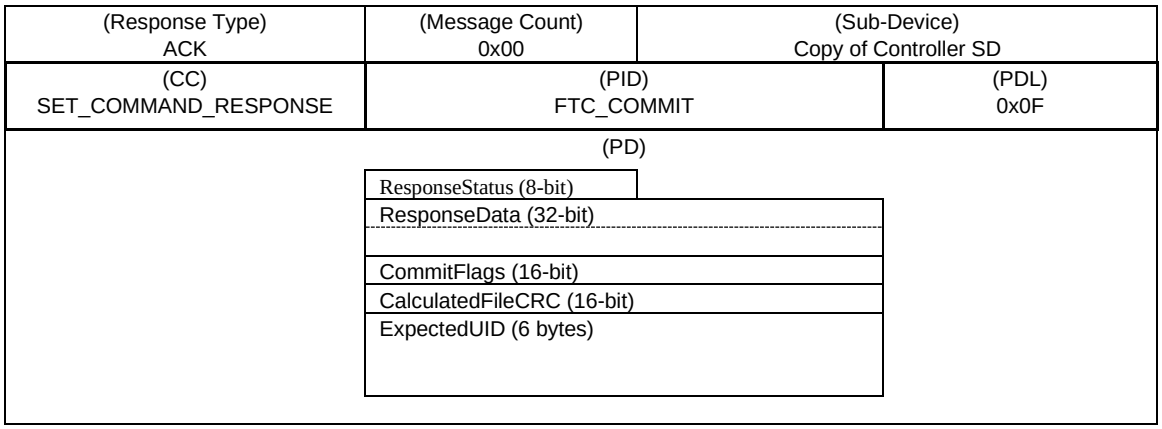
Controllers shall NOT use the value FTC_DEF_SESSIONID_ALL with this Parameter Message.

CommitFlags (16-bit):
Controllers shall set the FTC_TF_TESTMODE and FTC_TF_DOWNLOAD bits to match those sent in the SET:FTC_INITIATE as defined in Section 13.1.3.

CommitFlags bits are defined in Table A-5.

13.3.4 Response to SET:FTC_COMMIT

Responder: (SET)



Data Description:

ResponseStatus (8-bit):
A ResponseStatus of FTC_RS_STATUS_OK indicates that Commit is being processed.

A ResponseStatus of FTC_RS_STATUS_INPROGRESS indicates that Validation is not yet complete

A ResponseStatus of FTC_RS_FILE_CRC_ERROR indicates that Validation is complete, but that the calculated FileCRC does not match that offered during the SET:FTC_INITIATE. The Responder shall report the Received File CRC value (from the SET:FTC_INITIATE) in the upper 16 bits and the Calculated File CRC value in the lower 16 bits of the ResponseData field.

A ResponseStatus of FTC_RS_MODAL_ERROR shall be used if the CommitFlags do not match the TransferFlags received in the SET:FTC_INITIATE.

Other Status conditions may be reported in accordance with Section 12.1 and Table A-4 of this standard.

ResponseData (32-bit):
Responders returning a ResponseStatus of FTC_RS_STATUS_OK shall return a Commit Time (in ms) in the ResponseData Field.

The valid range is 0x00000000-0x00400000. (Max 4,194,304 ms or approx. 1.2hrs)

Responder shall return a value to indicate the maximum time the Responder may be offline and not able to communicate.

A Controller shall not attempt to communicate with the Responder during the Commit Time.

Controllers in receipt of a value out of range may wish to advise the user.

CommitFlags (16-bit):
Responders shall set this to the value offered as the CommitFlags in the SET_COMMAND.

CalculatedFileCRC:
The CRC that the Responder has calculated for the entire FTC Upload data received.

Responders able to calculate FileCRC shall check this value against the FileCRC received from the Controller in the SET:FTC_INITIATE Parameter Message. A ResponseStatus of FTC_RS_STATUS_OK indicates that CalculatedFileCRC matches the FileCRC received from the Controller.

A ResponseStatus of FTC_RS_FILE_CRC_ERROR indicates that CalculatedFileCRC does NOT match the FileCRC received from the Controller. The Responder shall report the Received File CRC value (from the SET:FTC_INITIATE) in the upper 16 bits and the Calculated File CRC value in the lower 16 bits of the ResponseData field.

Responders unable to calculate the CalculatedFileCRC shall set this field to 0x0000.

ExpectedUID:
The UID that the Responder expects to be using after the FTC Upload has been successfully completed.

Responders that do not expect the UID to change shall return the current UID.

Responders that expect the UID to change shall return the expected new UID.

Responders that expect the UID to change but do NOT know the new value shall return 0x000000000000.

Controllers should do a complete discovery if the expected UID does not respond after the Commit Time has elapsed.

Responders should attempt to ensure that the ExpectedUID after any firmware update is the same as the previous [RDM] UID. Controllers shall recognize that this is not always possible and should take steps to ensure that conflicts or UID duplications do not occur. If in doubt, it is recommended that Controllers do not proceed with the Commit.

13.4 Cancel Transfer (FTC_CANCEL)

Controllers wishing to cancel transfer of data may issue a SET:FTC_CANCEL and the file transfer process shall be abandoned by the target Responder.

After successful processing of a SET:FTC_CANCEL Parameter Message, a Responder shall return a SessionID of FTC_DEF_NO_SESSIONID_OFFERED to any subsequent GET:FTC_INITIATE Parameter Message.

Responders that have switched to a Bootloader or alternate application to process the file transfer should return to the previous application where possible. Responders that cannot return shall still support FTC Parameter Messages in order to allow the Controller to retry the transfer.

13.4.1 Controller: SET Request to Cancel: (SET:FTC_CANCEL)

Controller: (SET) *Controller Request to cancel transfer.*

(Port ID) 0x01-0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root)
(CC) SET_COMMAND	(PID) FTC_CANCEL	(PDL) 0x01
(PD) SessionID		

Data Description:

SessionID (8-bit):
Controllers shall set this to the SessionID relating to the transfer that they wish to cancel.

Controllers that wish to cancel a Responder from any current transfers, without reference to a specific SessionID shall set this field to FTC_DEF_SESSIONID_ALL, as defined in Table A-1.

13.4.2 Response to SET:FTC_CANCEL

Responder Response

(Response Type) ACK	(Message Count) 0x00	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_RESPONSE	(PID) FTC_CANCEL	(PDL) 0x05
(PD) ResponseStatus (8-bit) ResponseData (32-bit)		

Data Description:

ResponseStatus (8-bit):
Responders shall check the SessionID given in the SET_COMMAND, and if not FTC_DEF_SESSIONID_ALL, test against the SessionID given in the SET:FTC_INITIATE Parameter Message and return a ResponseStatus of FTC_RS_SESSIONID_MISMATCH if they are different.

A ResponseStatus of FTC_RS_STATUS_OK indicates that Cancel has been processed.

A ResponseStatus of FTC_RS_STATUS_IN_PROGRESS indicates that the Cancel is still being processed.

Responders shall treat Cancel as a priority and no other ResponseStatus values should be supported.

ResponseData (32-bit):
Responders returning a ResponseStatus of FTC_RS_STATUS_IN_PROGRESS shall set this field to return the expected remaining time in ms.

13.5 Obtain File List (FTC_FILELIST)

The FTC_FILELIST Parameter Message is used to retrieve File Information supported by the Responder.

13.5.1 Controller: Get request to obtain FileList (GET:FTC_FILELIST)

Controller: (GET)

(Port ID) 0x01-0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) FTC_FILELIST	(PDL) 0x00
(PD) Not Present		

13.5.2 Response to GET:FTC_FILELIST

This list is a packed list.

Responder: (GET Response)

(Response Type) ACK/ACK_OVERFLOW	(Message Count) 0x00	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) FTC_FILELIST	(PDL) 0x2B-0xD7
(PD) FileID (8-bit) Capabilities Bit Field (32-bit) File Description Text (Fixed 32 bytes) File Suffix Text (Fixed 6 bytes)		

Data Description:

FileID (8-bit):
The FileID field represents a unique defined identification for each file in the packed list. The valid range is greater than FTC_DEF_NO_FILEID_OFFERED and less than FTC_DEF_MULTIPLE_FILEID as defined in Table A-1.

Values of FTC_DEF_NO_FILEID_OFFERED and FTC_DEF_MULTIPLE_FILEID are reserved with special meaning and shall not be returned in this field.

The FileID shall be unique for different files supported by a given Responder model and may be different for different Responder models.

Capabilities Bit Field (32-bit):
Responders shall set this in accordance with Section 8.6.7 and as defined in Table A-7 of this standard.

A Controller should utilize the declared capabilities to optimize the operation of the transfer process, or inform the user of any limitations.

File Description Text (Fixed 32 byte field):
This field is a user defined description of the file content or use.

File Suffix Text (Fixed 6 byte field):
This field is a user defined extension that may be used to select the desired filename. Any suffix separator should not be included in the file suffix text.

13.6 Transfer Data from a Responder (FTC_TRANSFER_DOWNLOAD)

The FTC_TRANSFER_DOWNLOAD Parameter Message is used to transfer complete file data from a Responder to a Controller.

Responders should only expect a data transfer after receipt of the SET:FTC_INITIATE Parameter Message.

Responders shall reject a FTC_TRANSFER_DOWNLOAD Parameter Message using the ResponseStatus FTC_RS_MODAL_ERROR if it is received before a SET:FTC_INITIATE has been processed by the Responder.

Responders shall test the SessionID given in the GET Parameter Message against the SessionID given in the SET:FTC_INITIATE Parameter Message and return a ResponseStatus of FTC_RS_SESSIONID_MISMATCH if they are different.

13.6.1 Controller: Transfer Data from a Responder (GET:FTC_TRANSFER_DOWNLOAD)

Controllers shall send a GET:FTC_TRANSFER_DOWNLOAD Parameter Message with the command field set to FTC_TD_GET_NEXT_PACKET.

Controllers shall determine the number of file bytes received by examining the Parameter Data Length (PDL) field and subtracting the 6 bytes of the FTC_TRANSFER_DOWNLOAD response header.

While the Responder has more data to transfer it should return a ResponseStatus of FTC_RS_STATUS_OK.

Controllers shall repeat this process until a ResponseStatus of FTC_RS_TRANSFER_COMPLETE is returned.

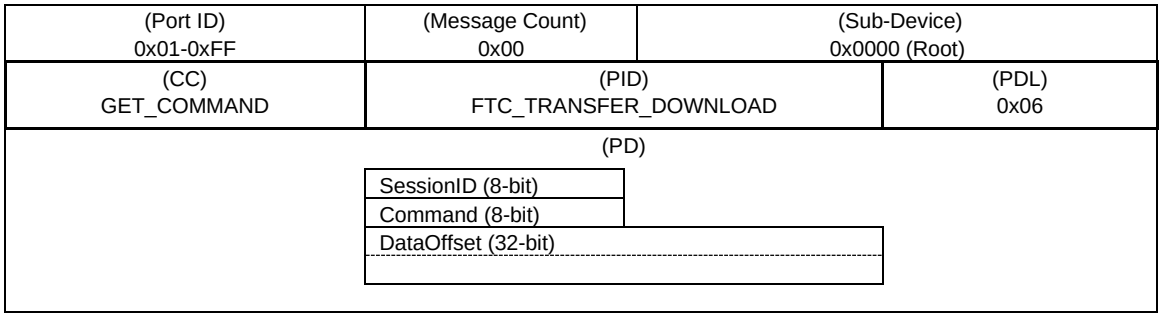
Controllers shall utilize the declared Inter-Packet Delay Time when sending successive GET:FTC_TRANSFER_DOWNLOAD Parameter Messages to a Responder.

If the Responder has declared support for PacketCRC generation, Controllers shall calculate the CRC on the received data and verify it matches the returned PacketCRC. If the PacketCRC mismatches, Controllers may attempt to re-request the packet using the FTC_TD_RESEND_LAST_PACKET command or abandon the transfer. A Controller shall limit re-request to a maximum of three attempts for any one packet.

If the Responder has declared support for FileCRC generation, a Controller shall request the FileCRC by sending a GET:FTC_TRANSFER_DOWNLOAD Parameter Message with the command field set to FTC_TD_GET_FILE_CRC. Controllers shall calculate the CRC on the whole file data received and verify it matches the returned FileCRC. If the FileCRC mismatches the Controller should discard all of the received data and restart the whole file download process. A Controller shall limit restarts to a maximum of three attempts for any one file.

Controllers may elect to abandon the transfer at any time.

Controller: (GET)



Data Description:

SessionID (8-bit):
The SessionID shall be created and issued by a Controller at the Initiate stage in accordance with Section 11 of this standard.

Controllers shall set this field to the SessionID generated at the time of the Initiate stage.

Command (8-bit):
Controllers shall set this to FTC_TD_GET_NEXT_PACKET to obtain the first and subsequent packets of the file from the Responder.

In the event of an error in the response from the Responder (for example CRC error) or low level [RDM] checksum error, a Controller may wish to re-request the packet using the command FTC_TD_RESEND_LAST_PACKET.

Controllers shall set this to FTC_TD_GET_FILE_CRC to obtain the FileCRC from Responders that have declared support for FTC_GENERATE_FILECRC in the Capabilities Bit Field.

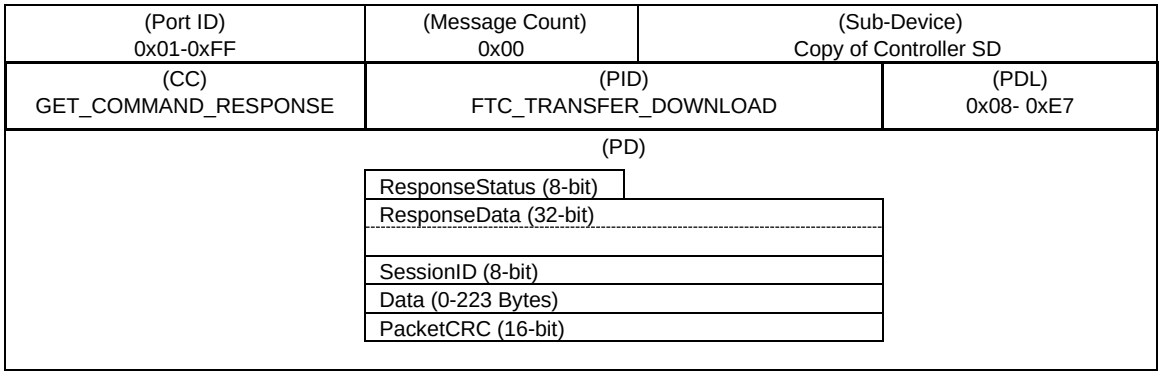
DataOffset (32-bit):
A Controller shall set this field to 0x00000000 when using FTC_TD_GET_NEXT_PACKET, FTC_TD_RESEND_LAST_PACKET or FTC_TD_GET_FILE_CRC.

This field is reserved for use by Controllers wishing to use Manufacturer Specific Transfer Download commands.

A Responder shall ignore this field unless utilizing such Manufacturer Specific commands.

13.6.2 Response to GET:FTC_TRANSFER_DOWNLOAD

Responder: (GET)



Responders shall reject the packet if the SessionID does not match that offered by the Controller during the Initiate stage.

Data Description:

ResponseStatus (8-bit):
A ResponseStatus of FTC_RS_STATUS_OK indicates that the last GET:FTC_TRANSFER_DOWNLOAD request was correctly received and that the Responder has more data available.

A ResponseStatus of FTC_RS_STATUS_INPROGRESS indicates that Responder is not yet ready to provide more data. The Responder shall report the remaining required time (in ms) in the ResponseData field.

A ResponseStatus of FTC_RS_TRANSFER_COMPLETE indicates that the Responder has provided all the available file data.

A ResponseStatus of FTC_RS_DOWNLOAD_COMMAND_ERROR indicates that the supplied Transfer Download command is not supported by the Responder.

A ResponseStatus of FTC_RS_DOWNLOAD_FILE_CRC indicates that the Responder has provided the calculated CRC for the whole file in the ResponseData in response to a FTC_TD_GET_FILE_CRC command, in accordance with Section 12.1.6 of this standard. Responders not supporting FileCRC shall respond with a ResponseStatus of FTC_RS_FILE_CRC_NOT_SUPPORTED.

Other Status conditions may be reported in accordance with Section 12.1 and Table A-4 of this standard.

ResponseData (32-bit):
Responders shall set this in accordance with Section 12.1 of this standard.

SessionID (8-Bit)
Responders shall set this in accordance with Section 11.1 of this standard.

Data (0-223 Bytes):
This field contains the data being transferred from the Responder to the Controller. The length of the data should be the TransferBlock Size, with the exception of the final packet where the length will be less than or equal to the TransferBlock Size.

Responders shall decide the number of data bytes from the requested file to be returned in each response packet and will set the Parameter Data Length (PDL) field accordingly.

PacketCRC (16-bit):
Responders that generate PacketCRC (as declared using the FTC_GENERATE_PACKETCRC capability) shall set the calculated CRC in the PacketCRC field in accordance with Section 10.4 of this standard.

The PacketCRC shall be calculated from the first byte of the Parameter Data for a length of Parameter Data Length minus 2.

Responders that do not generate PacketCRC shall set the PacketCRC field to 0x0000.

13.6.3 Packet Timing

Controllers shall use the Initial Delay Time values returned by the Responder during the INITATE stage to control the time between the SET:FTC_INITIATE Parameter Message and the first packet sent using the FTC_TRANSFER_DOWNLOAD Parameter Message.

Controllers shall use the Inter-Packet Delay Time value returned by the Responder during the INITATE stage to control the time between each packet request.

Accumulated Byte Count and Accumulated Byte Delay Time are not used for the FTC_TRANSFER_DOWNLOAD Parameter Message.

13.6.4 Download Cancel

A Controller wishing to abandon a Download shall use FTC_CANCEL in accordance with Section 13.4 of this standard.

Appendix A: Defined Parameters

Table A-1: General Defines

Define	Valid Value(s)	Controller	Responder
FTC_DEF_NO_FILEID_OFFERED	0x00	✓	
FTC_FILEID (*Valid range)*	0x01 – 0xFE	✓	✓
FTC_DEF_MULTIPLE_FILEID	0xFF		✓
FTC_DEF_NO_SESSIONID_OFFERED	0x00	✓	✓ Note1
FTC_SESSIONID (*Valid range)*	0x01 – 0xFE	✓	✓
FTC_DEF_SESSIONID_ALL	0xFF	✓	

Note 1: A Responder should only report FTC_DEF_NO_SESSIONID_OFFERED to a GET:FTC_INITIATE when there is no active Transfer session.

Table A-2: Parameter ID Defines

Note: Parameter Message PID values are not yet defined – final values will be assigned at time of formal release of the standard.

GET	SET	RDM Parameter ID's (Slot 20-21)	Value	Comment	Required
✓	✓	FTC_INITIATE			✓
✓	✓	FTC_TRANSFER_UPLOAD			✓
✓	✓	FTC_COMMIT			✓
	✓	FTC_CANCEL			✓
✓		FTC_FILELIST			Note 1
✓		FTC_TRANSFER_DOWNLOAD			

Note 1: Required if Responder supports multiple FileIDs

Table A-3: ResponseStatus Defines (GET/SET ResponseStatus)

ResponseStatus Defines	Value	Comments
FTC_RS_STATUS_OK	0x00	Last request was good
FTC_RS_INITOK_UL	0x01	Requirements Response to an UPLOAD request was good [Controller sends data TO Responder]
FTC_RS_INITOK_DL	0x02	Requirements Response to a DOWNLOAD request was good [Controller gets data FROM Responder]
FTC_RS_STATUS_IN_PROGRESS	0x03	The Responder is busy with the last requested action
FTC_RS_TRANSFER_COMPLETE	0x04	Upload or Download final packet transfer was good
FTC_RS_MODAL_ERROR	0x05	The Responder is in the wrong state to accept the command.
FTC_RS_SWITCH_TO_BOOTLOADER	0x06	The Responder is switching to Bootloader and requires the Controller to Delay and then re-issue the SET:FTC_INITIATE.
FTC_RS_SESSIONID_MISMATCH	0x07	SessionID does not match expected SessionID
FTC_RS_UNSUPPORTED_FILEID	0x08	The Responder does not support the requested/offered FileID
FTC_RS_FILE_NOT_COMPATIBLE	0x09	The file is not compatible with the Responder to Upload
FTC_RS_FILE_NOT_AVAILABLE	0x0A	The file is not currently available from Responder to Download
FTC_RS_PACKET_CRC_ERROR	0x0B	The Responder reports CRC mismatch on this packet
FTC_RS_FILE_CRC_ERROR	0x0C	The Responder reports CRC mismatch on this File
FTC_RS_VALIDATION_ERROR	0x0D	The Responder cannot validate the data uploaded by the Controller. The transfer has an invalid internal check, checksum or Manufacturer reason for rejection
FTC_RS_E137_LOCKACTIVE	0x10	Locked: Unlock using [PIDS-1]
FTC_RS_OTHER_LOCKACTIVE	0x11	Locked: Unlock using other scheme
FTC_RS_WRITE_PROTECT	0x12	A memory protection scheme is preventing acceptance of data. SD card may be write protected etc.
FTC_RS_INVALID_DIRECTION	0x13	The Responder does not support the requested transfer direction
FTC_RS_OFFSET_ERROR	0x14	Unexpected or out of range file offset
FTC_RS_FILESIZE_ERROR	0x15	Transfer data beyond declared File Size
FTC_RS_FTCVERSION_ERROR	0x16	The Responder cannot deal with FTCVERSION as offered by Controller
FTC_RS_FILE_CRC_NOT_SUPPORTED	0x17	FileCRC has been requested but the Responder does not support FileCRC
FTC_RS_DOWNLOAD_KEY_REQUIRED	0x18	Responder required a Download Key and no valid key was supplied
FTC_RS_DOWNLOAD_FILE_CRC	0x19	Download FileCRC Ok
FTC_RS_DOWNLOAD_COMMAND_ERROR	0x1A	Invalid Download Command Code
RESERVED	0x1B-0x7E	
FTC_RS_UNRESOLVED_ERROR	0x7F	Used for any error condition not defined above
MANUFACTURER_SPECIFIC	0x80-0xFF	Any undefined Status should be displayed as Unknown Status Response, together with the Status and Data

Table A-4: Use of ResponseStatus and ResponseData fields

ResponseStatus and ResponseData field use is described in Section 12.1

ResponseStatus	ResponseData Field use	GET:FTC_INITIATE	SET:FTC_INITIATE	GET:FTC_TRANSFER_UPLOAD	SET:FTC_TRANSFER_UPLOAD	GET:FTC_COMMIT	SET:FTC_COMMIT	GET:FTC_TRANSFER_DOWNLOAD	SET:FTC_CANCEL	GET:FTC_FILELIST
FTC_RS_STATUS_OK	0x00000000	Y		Y	Y			Y	Y	
	Communication Offline Delay					Y	Y			
FTC_RS_STATUS_IN_PROGRESS	Time to wait			Y	Y	Y	Y	Y	Y	
FTC_RS_INITOK_UL	0x00000000		Y							
FTC_RS_INITOK_DL	0x00000000		Y							
FTC_RS_SWITCH_TO_BOOTLOADER	Time to wait		Y							
FTC_RS_PACKET_CRC_ERROR	Upper 16 bits: Received CRC; Lower 16 bits: Calculated CRC		Y	Y	Y					
FTC_RS_FILE_CRC_ERROR	Upper 16 bits: Received CRC; Lower 16 bits: Calculated CRC					Y	Y			
FTC_RS_TRANSFER_COMPLETE	0x00000000			Y	Y			Y		
FTC_RS_UNSUPPORTED_FILEID	Upper 16 bits: 0x00 / Offered FileID; Lower 16 bits: 0x00 / Supported FileID	Y	Y							
FTC_RS_SESSIONID_MISMATCH	Upper 16 bits: Received SessionID; Lower 16 bits: Expected SessionID	Y	Y	Y	Y	Y	Y	Y	Y	
FTC_RS_MODAL_ERROR	Upper 16 bits: 0x0000; Lower 16 bits: Manufacturer Specific	Y	Y	Y	Y	Y	Y	Y		
FTC_RS_FILE_NOT_COMPATIBLE	Upper 16 bits: 0x0000; Lower 16 bits: Manufacturer Specific		Y	Y	Y	Y	Y			
FTC_RS_FILE_NOT_AVAILABLE	Upper 16 bits: 0x0000; Lower 16 bits: Manufacturer Specific		Y					Y		
FTC_RS_VALIDATION_ERROR	Upper 16 bits: 0x0000; Lower 16 bits: Manufacturer Specific					Y	Y			
FTC_RS_E137_LOCKACTIVE	Upper 16 bits: 0x0000; Lower 16 bits: Manufacturer Specific	Y	Y	Y	Y	Y	Y	Y		
FTC_RS_OTHER_LOCKACTIVE	Upper 16 bits: 0x0000; Lower 16 bits: Manufacturer Specific	Y	Y	Y	Y	Y	Y	Y		
FTC_RS_WRITE_PROTECT	Upper 16 bits: 0x0000; Lower 16 bits: Manufacturer Specific	Y	Y	Y	Y	Y	Y	Y		
FTC_RS_INVALID_DIRECTION	0x00000000		Y				Y			
FTC_RS_OFFSET_ERROR	Expected Offset			Y	Y			Y		
FTC_RS_FILESIZE_ERROR	Expected File Size		Y	Y	Y					
FTC_RS_FTCVERSION_ERROR	Upper 16 bits: Supported Major; Lower 16 bits: Supported Minor	Y	Y							
FTC_RS_DOWNLOAD_FILE_CRC	Upper 16 bits: 0x0000; Lower 16 bits: Calculated CRC from Responder							Y		
FTC_RS_FILE_CRC_NOT_SUPPORTED	0x00000000							Y		
FTC_RS_DOWNLOAD_KEY_REQUIRED	0x00000000		Y							
FTC_RS_DOWNLOAD_COMMAND_ERROR	Return command as Received							Y		
FTC_RS_UNRESOLVED_ERROR	Upper 16 bits: 0x0000; Lower 16 bits: Manufacturer Specific	Y	Y	Y	Y	Y	Y	Y		
FTC_RS_MANUFACTURER_SPECIFIC	Upper 16 bits: 0x0000; Lower 16 bits: Manufacturer Specific	Y	Y	Y	Y	Y	Y	Y		

Table A-5: TransferFlags/CommitFlags Defines

TransferFlags/ CommitFlags	Bit Mask	Comment
FTC_TF_TESTMODE	0x0001	SET if this is a test
FTC_TF_DOWNLOAD	0x0002	SET if this is a download
FTC_TF_RESERVED1	0x0004	
FTC_TF_RESERVED2	0x0008	
FTC_TF_RESERVED3	0x0010	
FTC_TF_RESERVED4	0x0020	
FTC_TF_RESERVED5	0x0040	
FTC_TF_RESERVED6	0x0080	
FTC_TF_RESERVED7	0x0100	
FTC_TF_RESERVED8	0x0200	
FTC_TF_RESERVED9	0x0400	
FTC_TF_MANUFACTURER_SPECIFIC1	0x0800	Reserved for Manufacturer Specific use.
FTC_TF_MANUFACTURER_SPECIFIC2	0x1000	Reserved for Manufacturer Specific use.
FTC_TF_MANUFACTURER_SPECIFIC3	0x2000	Reserved for Manufacturer Specific use.
FTC_TF_MANUFACTURER_SPECIFIC4	0x4000	Reserved for Manufacturer Specific use.
FTC_TF_MANUFACTURER_SPECIFIC5	0x8000	Reserved for Manufacturer Specific use.

Table A-6: Transfer Download Command Defines

Transfer Download Command	Value	Comment
RESERVED	0x00	
FTC_TD_GET_NEXT_PACKET	0x01	Responder replies with file data at next Responder offset
FTC_TD_RESEND_LAST_PACKET	0x02	Responder replies with file data at the last Responder offset returned.
FTC_TD_GET_FILE_CRC	0x03	Controller sends after Responder provides a ResponseStatus of transfer complete. Responder replies with FileCRC for the last transferred file
RESERVED	0x04-0x7F	
MANUFACTURER_SPECIFIC	0x80-0xFF	

Table A-7: Responder Capabilities Defines

Responder Capabilities	Bit Mask	Comments
FTC_ACCEPT_UPLOAD	0x00000001	This shall be set if Responder is able to accept offered FileID for upload.
FTC_ACCEPT_DOWNLOAD	0x00000002	This shall be set if Responder is able to accept offered FileID for download.
FTC_PROCESS_FILECRC	0x00000004	Set if Responder processes FileCRC on upload
FTC_PROCESS_PACKETCRC	0x00000008	Set if Responder processes PacketCRC.
FTC_GENERATE_FILECRC	0x00000010	Set if Responder generates FileCRC on download.
FTC_GENERATE_PACKETCRC	0x00000020	Set if Responder generates PacketCRC on download.
FTC_DOWNLOAD_KEY	0x00000040	Set if a Download Key is required.
FTC_BOOTLOADER_SWITCH	0x00000080	Set if Responder will switch from normal to Bootloader functionality to further process FTC for this FileID.
FTC_NSC_NO_INTERLEAVE	0x00000100	Set if interleaving of NULL START Code DMX and/or non-FTC RDM Data during FTC transfer may result in Responder failure.
FTC_NSC_IGNORE	0x00000200	Set if NULL START Code DMX packets are ignored during transfers
FTC_FUNCTIONAL_LIMIT	0x00000400	Set if the Responder limits normal functions during transfers.
Reserved1	0x00000800	Reserved.
FTC_E137_LOCK	0x00001000	Set if [PIDS-1] PID Unlock required before upload accepted.
FTC_OTHER_LOCK	0x00002000	Set if Proprietary Unlock required before upload accepted.
Reserved2	0x00004000	Reserved.
FTC_ACCEPT_BROADCASTS	0x00008000	Set if Responder accepts E1.37-4 Parameter Messages as [RDM] ALL_DEVICES_ID Manufacturer Broadcast
FTC_FAIL_MAY_BRICK	0x00010000	Set if failure to successfully complete the transfer may cause the Responder to become inoperative.
FTC_ALTERNATE_ERROR_RECOVERY	0x00020000	Set if Manufacturer is declaring support for some alternate mechanism of fatal error recovery. Such methods may involve processes not described by this standard such as USB uploads, media replacement, power cycling or other software solutions.
Reserved3	0x00040000	Reserved.
Reserved4	0x00080000	Reserved.
Reserved5	0x00100000	Reserved.
Reserved6	0x00200000	Reserved.
Reserved7	0x00400000	Reserved.
FTC_TESTMODE_SUPPORTED	0x00800000	Responder supports use of Testmode
Reserved8	0x01000000	Reserved.
Reserved9	0x02000000	Reserved.
Reserved10	0x04000000	Reserved.
Manufacturer Specific1	0x08000000	Reserved for Manufacturer Specific use.
Manufacturer Specific2	0x10000000	Reserved for Manufacturer Specific use.
Manufacturer Specific3	0x20000000	Reserved for Manufacturer Specific use.
Manufacturer Specific4	0x40000000	Reserved for Manufacturer Specific use.
Manufacturer Specific5	0x80000000	Reserved for Manufacturer Specific use.

Appendix B: Control Flow – Firmware Upload

Example B-1 : File Upload Controller to Responder (Single FileID supported)

Controller sends SET:FTC_INITIATE Responder replies with FTC_RS_INITOK_UL and the Transfer Requirements
Controller waits Initial Delay Time
Controller sends SET:FTC_TRANSFER_UPLOAD (Note 1) Responder replies with FTC_RS_STATUS_OK (Note 2)
Controller waits declared Inter-Packet Delay Time (Note 3)`
Controller sends SET:FTC_TRANSFER_UPLOAD (Note 1) Responder replies with FTC_RS_STATUS_OK
Controller waits declared Inter-Packet Delay Time (Note 3)
Repeat as required
Controller sends SET:FTC_TRANSFER_UPLOAD for the final packet (Note 1) Responder replies with FTC_RS_TRANSFER_COMPLETE (Note 4)
Responder commences Validation
Controller waits Validation Delay Time (Note 5)
Controller sends SET:FTC_COMMIT Responder replies with FTC_RS_STATUS_OK and Commit Time
Controller waits returned Commit Time
Controllers may assume that the FTC process is complete after the expiry of the Commit Time (Note 6)

Note 1: The TransferBlock Size is as determined in the response to the SET:FTC_INITIATE Parameter Message.

Note 2: The DataOffset field in the reply with the FTC_RS_STATUSOK indicates the next expected DataOffset. The ResponseData field is set to zero.

Note 3: The time between packets may also be extended after a series of packets has been transferred, as defined by the Accumulated Byte Count and Accumulated Byte Delay Time.

Note 4: The DataOffset returns the File Size received. The ResponseData field is set to zero.

Note 5: The Validation Delay Time is as determined in the response to the SET:FTC_INITIATE Parameter Message.

Note6: The Controller should assume that the Responder may have new features. Controller may wish to issue a [RDM] GET:DEVICE_INFO Parameter Message to determine the new software version.

Example B-2 : File Upload Controller to Responder (Multiple FileIDs)

Controller sends GET:FTC_INITIATE or SET:FTC_INITIATE with TransferFlags FTC_TF_DOWNLOAD bit CLEAR Responder replies with FTC_RS_STATUS_OK and FileID of FTC_DEF_MULTIPLE_FILEID to indicate that multiple files are supported
Controller sends GET:FTC_FILELIST Responder returns list of FileIDs, names and capabilities
Controller sends SET:FTC_INITIATE for required FileID with TransferFlags FTC_TF_DOWNLOAD bit CLEAR Responder replies with FTC_RS_INITOK_UL and the Transfer Requirements
Controller waits Initial Delay Time
Controller sends SET:FTC_TRANSFER_UPLOAD (Note 1) Responder replies with FTC_RS_STATUS_OK
Controller waits declared Inter-Packet Delay Time (Note 2)
Controller sends SET:FTC_TRANSFER_UPLOAD (Note 1) Responder replies with FTC_RS_STATUS_OK
Controller waits declared Inter-Packet Delay Time (Note 2)
Repeat as required
Controller sends SET:FTC_TRANSFER_UPLOAD for the final packet Responder replies with FTC_RS_TRANSFER_COMPLETE (Note 4).
Responder commences Validation
Controller waits Validation Delay Time (Note 3)
Controller sends SET:FTC_COMMIT Responder replies with FTC_RS_STATUS_OK and Commit Time
Controller waits returned Commit Time
Controllers may assume that the FTC process is complete after the expiry of the Commit Time (Note 5).

Note 1: The TransferBlock Size is as determined in the response to the SET:FTC_INITIATE Parameter Message.

Note 2: The time between packets may also be extended after a series of packets has been transferred, as defined by the Accumulated Byte Count and Accumulated Byte Delay Time.

Note 3: The Validation Delay Time is as determined in the response to the SET:FTC_INITIATE Parameter Message.

Note 4: The DataOffset returns the File Size received. The ResponseData field is set to zero.

Note 5: The Controller should assume that the Responder may have new features. Controller may wish to attempt a [RDM] GET:DEVICE_INFO Parameter Message to determine the new software version.

Example B-3 : File Upload Controller to Responder (Responder needs more time to check packets)

Controller sends SET:FTC_INITIATE with TransferFlags FTC_TF_DOWNLOAD bit CLEAR Responder replies with FTC_RS_INITOK_UL and the Transfer Requirements
Controller waits Initial Delay Time
Controller sends SET:FTC_TRANSFER_UPLOAD (Note 1) Responder replies with FTC_RS_IN_PROGRESS and Time required (Note 2)
Controller waits Time required (Note 2)
Controller optionally sends GET:FTC_TRANSFER_UPLOAD Responder replies with FTC_RS_STATUS_OK if ready to continue Or Responder replies with FTC_RS_IN_PROGRESS and remaining Time required (Note 2)
Controller sends SET:FTC_TRANSFER_UPLOAD (Note 1) Responder replies with FTC_RS_STATUS_OK
Controller waits declared Inter-Packet Delay Time (Note 3)
Repeat as required
Controller sends SET:FTC_TRANSFER_UPLOAD for the final packet (Note 1) Responder replies with FTC_RS_TRANSFER_COMPLETE (Note 4)
Responder commences Validation
Controller waits Validation Delay Time (Note 3)
Controller sends SET:FTC_COMMIT Responder replies with FTC_RS_STATUS_OK and Commit Time
Controllers may assume that the FTC process is complete after the expiry of the Commit Time (Note 5).

Note 1: The TransferBlock Size is as determined in the response to the SET:FTC_INITIATE Parameter Message.

Note 2: The ResponseStatus field in the reply with the ResponseStatus FTC_RS_IN_PROGRESS indicates required Delay.

Note 3: The time between packets may also be extended after a series of packets has been transferred, as defined by the Accumulated Byte Count and Accumulated Byte Delay Time.

Note 4: The DataOffset returns the File Size received. The ResponseData field is set to zero.

Note 5: The Controller should assume that the Responder may have new features. Controller may wish to attempt a [RDM] GET:DEVICE_INFO Parameter Message to determine the new software version.

Example B-4 : Controller to Responder: File Download

Controller sends SET:FTC_INITIATE with TransferFlags FTC_TF_DOWNLOAD bit SET Responder replies with FTC_RS_INITOK_DL and the Transfer Requirements
Controller waits Initial Delay Time (Note 1)
Controller sends GET:FTC_TRANSFER_DOWNLOAD with the Transfer Download Command FTC_TD_GET_NEXT_PACKET Responder replies with FTC_RS_STATUS_OK and the packet data
Controller waits declared Inter-Packet Delay Time (Note 1)
Repeat as required until final packet
Controller sends GET:FTC_TRANSFER_DOWNLOAD with Transfer Download Command FTC_TD_GET_NEXT_PACKET for final packet Responder replies with FTC_RS_TRANSFER_COMPLETE and the packet data
Controller waits Validation Delay Time (Note 3)
Controller sends GET:FTC_TRANSFER_DOWNLOAD with the Transfer Download Command FTC_TD_DOWNLOAD_FILE_CRC. (Note 4) Responder replies with FTC_RS_DOWNLOAD_FILE_CRC and the File CRC (Note 5)
Controller calculates the CRC on the received data and verifies with received File CRC (Note 6)

Note 1: A Responder performing File Download may require an Initial Delay Time due to internal processing to access the required data file.

Note 2: A Responder performing a File Download may require an Inter-Packet Delay in order for it to pre-calculate the PacketCRC or for other internal processing..

Note 3: The Validation Delay Time is as determined in the response to the SET:FTC_INITIATE Parameter Message.

Note 4: The ability of the Responder to provide a FileCRC is declared as part of the Responder Capabilities returned in the SET:FTC_INITIATE response. A Controller should only request the FileCRC when the Responder is able to provide it.

Note 5: The calculated CRC is provided in the ResponseData field.

Note 6: If the retrieved File CRC does not match that calculated by the Controller, the file data is discarded.
A Controller may wish to restart the download from the beginning of the file by issuing another SET:FTC_INITIATE Parameter Message.

Example B-5 : File Download Responder to Controller (Multiple FileIDs)

Controller sends GET:FTC_INITIATE or SET:FTC_INITIATE with TransferFlags FTC_TF_DOWNLOAD bit SET Responder replies with FTC_RS_STATUS_OK and FileID of FTC_DEF_MULTIPLE_FILEID to indicate that multiple files are supported
Controller sends GET:FTC_FILELIST Responder returns list of FileIDs, names and capabilities
Controller sends SET:FTC_INITIATE for required FileID with TransferFlags FTC_TF_DOWNLOAD bit SET Responder replies with FTC_RS_INITOK_DL and the Transfer Requirements
Controller waits Initial Delay Time (Note 1)
Controller sends GET:FTC_TRANSFER_DOWNLOAD with Transfer Download Command FTC_TD_GET_NEXT_PACKET Responder replies with FTC_RS_STATUS_OK and the packet data
Controller waits declared Inter-Packet Delay Time (Note 2)
Repeat as required until final packet
Controller sends GET:FTC_TRANSFER_DOWNLOAD with Transfer Download Command FTC_TD_GET_NEXT_PACKET for final packet Responder replies with FTC_RS_TRANSFER_COMPLETE and the packet data
Controller waits Validation Delay Time (Note 3)
Controller sends GET:FTC_TRANSFER_DOWNLOAD with Transfer Download Command FTC_TD_GET_FILE_CRC (Note 4) Responder replies with FTC_RS_STATUS_OK and File CRC (Note 5)
Controller calculates the CRC on the received data and verifies with received File CRC (Note 6)

Note 1: A Responder performing File Download may require an Initial Delay Time due to internal processing to access the required data file.

Note 2: A Responder performing a File Download may require an Inter-Packet Delay in order for it to pre-calculate the Packet CRC or for other internal processing..

Note 3: The Validation Delay Time is as determined in the response to the SET:FTC_INITIATE Parameter Message.

Note 4: The ability of the Responder to provide a FileCRC is declared as part of the Responder Capabilities returned in the SET:FTC_INITIATE response. A Controller should only request the FileCRC when the Responder is able to provide it.

Note 5: The calculated CRC is provided in the ResponseData field.

Note 6: If the retrieved File CRC does not match that calculated by the Controller, the file data is discarded.
A Controller may wish to restart the download from the beginning of the file by issuing another SET:FTC_INITIATE Parameter Message.

Appendix C: Transfer Time Guidance

This analysis is provided as informative. All calculations are approximate.

It is intended to provide guidance as to the suitability of the transfer process, especially as it might relate to larger files.

To transfer a file of 64Kbytes of data will require [65,535 ÷ 224 = 293] packets to be sent.

Assuming the best case of Break, MAB and inter-character time in the outgoing transmission, each transmitted packet takes:

*176 µs Break + 12us MAB+((25 + 231) * 44)µs [11.45ms].*

Assuming each packet gets an ACK, with a PDL of 9 bytes, the response takes:

*176 µs Break + 12us MAB+((25 + 9) * 44)µs [1.68ms].*

This assumes the best case of Break, MAB and inter-character time in the return (response) transmission.

*Assuming best case turn-around of 176 µs there is 293 * 176 µs [approximately 51.6ms] of system delay:*

*Best case Total transfer time estimated at (293 * 11.45ms Tx Data) [3,354.85ms] + (293 * 1.68ms Rx Data) [492.24ms]+ System delay [51.6ms] which gives approximately 3,898.69ms.*

*Assuming worst case turnaround of 2.8ms there is 292 * 2.8ms [817.6ms] of system delay, so the transfer time would increase to 4,716.29ms.*

Further delays may occur depending on the time taken by the Responder to validate transfers and reprogram memory.

If we take the above values and assume a transfer file size of 1Mbyte, the amount of packets to be transmitted would be 4,688 and the expected duration would be 61,160.16 ms.

	64kb	**1024kb**	**2048kb**
Packets	293	4,688	9,376
Duration [ms]	4,719.29	75,508.64	151,017.28

Note: *A file size of 4GB would, however, take almost 3 days using this protocol. The use of this standard to transfer files of this size may not be appropriate for some applications.*

Appendix D: CRC Algorithm and Example

A CRC is called an *n*-bit CRC when its check value is *n*-bits. For a given *n*, multiple CRCs are possible, each with a different polynomial. This standard uses a 16-bit CRC commonly known as CRC-16-ANSI, CRC-16-IBM, or Modbus CRC using the polynomial x16+x15+x2+1.

The Cyclical Redundancy Check (CRC) field is two bytes, containing a 16bit binary value.

The CRC value is calculated by the Controller.

The Responder recalculates a CRC during receipt of the message and compares the calculated value against the actual value it received in the CRC field. If the two values are not equal, an error results.

CRC Generation

The CRC is created by preloading a 16-bit register to all 1's. Then a process begins of applying successive 8-bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC.

During generation of the CRC, each 8-bit character is exclusive ORed with the register contents. The result is then shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value. If the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8-bit character is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final contents of the register, after all the characters of the message have been applied, is the CRC value.

A procedure for generating a CRC is:

1) Load a 16-bit register with 0xFFFF (all 1's). Call this the CRC register.
2) Exclusive OR the first 8-bit byte of the message with the low–order byte of the 16-bit CRC register, putting the result in the CRC register.
3) Shift the CRC register one bit to the right (toward the LSB), setting the MSB to zero. Extract and examine the LSB.
4) If the LSB is 0, repeat Step 3 (another shift). If the LSB is 1, exclusive OR the CRC register with the polynomial value 0xA001 (1010 0000 0000 0001).
5) Repeat Steps 3 and 4 until 8 shifts have been performed. When this is done, a complete 8-bit byte will have been processed.
6) Repeat Steps 2 through 5 for the next 8-bit byte of the message. Continue doing this until all bytes have been processed.
7) The CRC register now contains the CRC value.

An example of a C language function performing CRC generation is shown below. All of the possible CRC values are preloaded into two arrays, which are simply indexed as the function increments through the message buffer.

One array contains all of the 256 possible CRC values for the high byte of the 16bit CRC field, and the other array contains all of the values for the low byte.

Indexing the CRC in this way provides faster execution than would be achieved by calculating a new CRC value with each new character from the message buffer.

CRC Generation Function

The function takes two arguments:

```
unsigned char *puchMsg  ; //pointer to the message buffer containing binary data to be used for
                            generating the CRC
unsigned long ulDataLen ; //The quantity of bytes in the message buffer.
```

The function returns the CRC as a type unsigned short.

```
unsigned short CRC16(*puchMsg, ulDataLen)
{
unsigned char uchCRCHi = 0xFF ; /* high byte of CRC initialized */
unsigned char uchCRCLo = 0xFF ; /* low byte of CRC initialized */
unsigned uIndex ;               /* will index into CRC lookup table */

while (ulDataLen--) /* pass through message buffer */
{
uIndex   = uchCRCHi ^ *puchMsgg++ ; /* calculate the CRC */
uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;
uchCRCLo = auchCRCLo[uIndex] ;
}
return (uchCRCHi << 8 | uchCRCLo) ;
}
```

High-Order Byte Table

/* Table of CRC values for high–order byte */

```
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40
} ;
```

Low-Order Byte Table

/* Table of CRC values for low–order byte */

```
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
0x40
} ;
```