



ANSI E1.20 - 2010
Entertainment Technology—RDM
Remote Device Management
Over DMX512 Networks

Document number: CP/2009-1017r2
© 2017 ESTA

Copyright © 2017 by Entertainment Service and Technology Association

[blank page]

Copyright Clearance Center



ANSI E1.20 - 2010 Entertainment Technology—RDM Remote Device Management Over DMX512 Networks

Copyright 2017 ESTA. All rights reserved.
CP/2009-1017r2

Approved as an American National Standard by the ANSI Board of Standards
Review on 4 January 2011.

This standard was originally published when the Entertainment Services and Technology Association was operating under the name of PLASA North America. ESTA has reverted to its original name, and this document has been rebranded with the current corporate name and logo. No changes have been made to the contents of the standard.

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 guarantees 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 his or her 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
630 Ninth Avenue, Suite 609
New York, NY 10036
USA
Phone: 1-212-244-1505
Fax: 1-212-244-1502
Email: standards@esta.org

The ESTA 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 within the entertainment industry by creating recommended practices and standards, monitoring standards issues around the world on behalf 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 USITT and VPLT, as well as representing the interests of ESTA members to ANSI, UL, and the NFPA. The Technical Standards Program is accredited by the American National Standards Institute.

The Technical Standards Council (TSC) was established to oversee and coordinate the Technical Standards Program. Made up of individuals experienced in standards-making work from throughout our industry, the Council 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 Council and its working groups as well as maintain a "Standards Watch" on behalf of members. Working groups include: Camera Cranes, Control Protocols, Electrical Power, Event Safety, Floors, Fog and Smoke, Followspot Position, Photometrics, Rigging, and Stage Lifts.

ESTA encourages active participation in the Technical Standards Program. There are several ways to become involved. If you would like to become a member of an existing working group, as have over four hundred people, you must complete an application which is available from the ESTA office. Your application is subject to approval by the working group and you will be required to actively participate in the work of the group. This includes responding to letter ballots and attending meetings. Membership in ESTA is not a requirement. You can also become involved by requesting that the TSC develop a standard or a recommended practice in an area of concern to you.

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.

Table of Contents

Introduction	1
Overview.....	1
1 Normative References	3
2 Physical Layer.....	4
2.1 General.....	4
2.2 Electrical Specifications and Physical Layer Overview	4
2.2.1 ANSI E1.11 - EF1.0	4
2.3 Termination Requirements	4
2.4 Maintaining data link state between packets	5
2.4.1 Line Bias Networks	5
2.5 Command port reference circuit	6
2.5.1 Details of the reference circuit.	6
2.6 Default line state control	7
3 Timing	8
3.1 Controller Timing Requirements.....	8
3.1.1 Controller Packet Timing.....	8
3.1.2 Controller Packet spacing	9
3.1.3 Driver shutoff time	10
3.2 Responder Timing Requirements	10
3.2.1 Responder Packet Timings.....	10
3.2.2 Responder Packet spacing	10
3.2.3 Responder discovery response driver enable time	11
3.2.4 Driver shutoff time	11
3.2.5 Discovery Response MARK time.....	11
3.3 Data collisions.....	12
4 In-Line Devices	12
4.1 Overview	12
4.1.1 Transparent In-Line devices	12
4.2 Transparent In-Line Device Timing Requirements	12
4.2.1 Port Turnaround	13
4.2.2 Data Delay	13
4.2.3 Bit Distortion.....	13
4.2.4 BREAK timing	14
4.3 In-Line devices operating as part of Non-RDM system	14
4.4 In-line Devices as Responders.....	14
5 Device Addressing.....	15
5.1 General.....	15
5.2 UID Text Representation.....	15

5.3 Broadcast Message Addressing	16
5.4 Responders with multiple Responder Ports in a single Device.....	16
6 Message Structure.....	17
6.1 Byte Ordering	17
6.2 Packet Format	17
6.2.1 START Code.....	18
6.2.2 Sub START Code	18
6.2.3 Message Length	18
6.2.4 Destination UID	19
6.2.5 Source UID.....	19
6.2.6 Transaction Number (TN)	19
6.2.7 Port ID / Response Type.....	19
6.2.8 Message Count.....	20
6.2.9 Sub-Device Field.....	22
6.2.10 Message Data Block (MDB).....	22
6.2.11 Checksum	24
6.3 Response Type Field Values	25
6.3.1 Acknowledge (RESPONSE_TYPE_ACK)	25
6.3.2 Acknowledge Overflow (RESPONSE_TYPE_ACK_OVERFLOW)	25
6.3.3 Acknowledge Timer (RESPONSE_TYPE_ACK_TIMER).....	27
6.3.4 Negative Acknowledge (RESPONSE_TYPE_NACK_REASON)	29
7 Discovery Method.....	30
7.1 General.....	30
7.2 Binary Search Tree	30
7.3 Discovery Process Steps	31
7.4 Discovery Mute Flag	31
7.4.1 Clearing of Mute Flag.....	31
7.5 Discovery Unique Branch Message (DISC_UNIQUE_BRANCH)	32
7.5.1 Response Unique Branch Message Decoding by Controller	34
7.5.2 Collisions.....	34
7.6 Discovery Mute/Un-Mute Messages.....	34
7.6.1 Control Field.....	34
7.6.2 Binding UID.....	35
7.6.3 Discovery Mute Message (DISC_MUTE)	35
7.6.4 Discovery Un-Mute Message (DISC_UN_MUTE)	36
7.7 Discovery Algorithm	37
7.8 Ongoing Device Discovery	39
8 Proxy Devices	40
8.1 General.....	40
8.2 Discovery.....	40
8.2.1 Represented Devices.....	40
8.2.2 Proxy Management.....	40

8.3 Response Messages	40
8.4 Proxy Management Messages	41
8.4.1 Get Proxied Device Count (PROXIED_DEVICE_COUNT)	41
8.4.2 Get Proxied Devices (PROXIED_DEVICES).....	42
9 Sub-Devices	43
9.1 General.....	43
9.2 Sub-Device Messages.....	43
9.2.1 Sub-Device Field.....	43
9.2.2 Using Sub-Devices	43
9.2.3 Required Sub-Device Messages	43
10 RDM Parameter Messages	44
10.1 Text Field Handling	44
10.2 Network Management Messages	44
10.2.1 Communication Status (COMMS_STATUS)	44
10.3 Collection of Queued and Status Messages	46
10.3.1 Get Queued Message (QUEUED_MESSAGE)	46
10.3.2 Get Status Messages (STATUS_MESSAGES).....	49
10.3.3 Get Status ID Description (STATUS_ID_DESCRIPTION)	52
10.3.4 Clear Status ID (CLEAR_STATUS_ID)	52
10.3.5 Get/Set Sub-Device Status Reporting Threshold (SUB_DEVICE_STATUS_REPORT_THRESHOLD)	53
10.4 RDM Information Messages	54
10.4.1 Get Supported Parameters (SUPPORTED_PARAMETERS)	54
10.4.2 Get Parameter Description (PARAMETER_DESCRIPTION).....	56
10.5 Product Information Messages	58
10.5.1 Get Device Info (DEVICE_INFO).....	58
10.5.2 Get Product Detail ID List (PRODUCT_DETAIL_ID_LIST)	61
10.5.3 Get Device Model Description (DEVICE_MODEL_DESCRIPTION).....	62
10.5.4 Get Manufacturer Label (MANUFACTURER_LABEL)	62
10.5.5 Get/Set Device Label (DEVICE_LABEL)	63
10.5.6 Get/Set Factory Defaults (FACTORY_DEFAULTS)	64
10.5.7 Get Language Capabilities (LANGUAGE_CAPABILITIES).....	65
10.5.8 Get/Set Language (LANGUAGE)	66
10.5.9 Get Software Version Label (SOFTWARE_VERSION_LABEL)	67
10.5.10 Get Boot Software Version ID (BOOT_SOFTWARE_VERSION_ID).....	68
10.5.11 Get Boot Software Version Label (BOOT_SOFTWARE_VERSION_LABEL).....	69
10.6 DMX512 Setup Messages	69
10.6.1 Get/Set DMX512 Personality (DMX_PERSONALITY)	69
10.6.2 Get DMX512 Personality Description (DMX_PERSONALITY_DESCRIPTION).....	71
10.6.3 Get/Set DMX512 Starting Address (DMX_START_ADDRESS)	72
10.6.4 Get Slot Info (SLOT_INFO).....	73
10.6.5 Get Slot Description (SLOT_DESCRIPTION)	74
10.6.6 Get Default Slot Value (DEFAULT_SLOT_VALUE)	75
10.7 Sensor Parameter Messages.....	75
10.7.1 Get Sensor Definition (SENSOR_DEFINITION)	76

10.7.2 Get/Set Sensor (SENSOR_VALUE)	78
10.7.3 Record Sensors (RECORD_SENSORS)	80
10.8 Power/Lamp Setting Parameter Messages	81
10.8.1 Get/Set Device Hours (DEVICE_HOURS)	81
10.8.2 Get/Set Lamp Hours (LAMP_HOURS)	82
10.8.3 Get/Set Lamp Strikes (LAMP_STRIKES)	83
10.8.4 Get/Set Lamp State (LAMP_STATE)	84
10.8.5 Get/Set Lamp On Mode (LAMP_ON_MODE)	85
10.8.6 Get/Set Device Power Cycles (DEVICE_POWER_CYCLES)	86
10.9 Display Setting Parameter Messages	87
10.9.1 Get/Set Display Invert (DISPLAY_INVERT)	87
10.9.2 Get/Set Display Level (DISPLAY_LEVEL)	88
10.10 Device Configuration Parameter Messages	89
10.10.1 Get/Set Pan Invert (PAN_INVERT)	89
10.10.2 Get/Set Tilt Invert (TILT_INVERT)	90
10.10.3 Get/Set Pan/Tilt Swap (PAN_TILT_SWAP)	91
10.10.4 Get/Set Device Real-Time Clock (REAL_TIME_CLOCK)	92
10.11 Device Control Parameter Messages	93
10.11.1 Get/Set Identify Device (IDENTIFY_DEVICE)	93
10.11.2 Reset Device (RESET_DEVICE)	94
10.11.3 Get/Set Power State (POWER_STATE)	95
10.11.4 Get/Set Perform Self Test (PERFORM_SELFTEST)	96
10.11.5 Get Self Test Description (SELF_TEST_DESCRIPTION)	97
10.11.6 Capture Preset (CAPTURE_PRESET)	98
10.11.7 Get/Set Preset Playback (PRESET_PLAYBACK)	99
11 Operational Issues	101
11.1 Polling Intervals	101
12 Protocol Support Issues	101
Appendix A: Defined Parameters (Normative)	102
Appendix B: Status Message ID's (Normative)	118
Appendix C: Slot Info (Normative)	120
Appendix D: Definitions (Normative)	122
Appendix E: Discovery Pseudo-Code Example (Informative)	126
E.1 Find Devices Function Call	126
E.2 Find Devices Pseudo-Code	126
Appendix F: Qualification tests for transmitter/receiver circuits used in RDM systems (Normative)	129
F.1 Qualification Tests for Command port Transmitter Circuits	129
F.1.1 Notes on Figure F-1	129
F.2 Output tests for testing transmitters / line bias networks for RDM command ports	130
F.3 Line loading tests for command ports	130

F.4 Notes on the responder test circuit	132
F.5 Testing Responder Transmitters	132
Contact Information.....	133
Acknowledgments:	134

List of Tables

Table 2-1: Command Port Reference Circuit Values	7
Table 3-1: Controller Packet Timing	8
Table 3-2: Controller Packet Spacing Times.....	9
Table 3-3: Responder Packet Timing	10
Table 3-4: Responder Packet Spacing Times.....	11
Table 5-1: Unique ID (UID) Format.....	15
Table 6-1: Byte Ordering.....	17
Table 6-2: Packet Format	17
Table 6-3: Example of Packet showing Message Length Pointer to Checksum	18
Table 6-4: Message Data Block (MDB) Format	22
Table 6-5: Command Class Description	22
Table 6-6: Checksum Usage Example.....	24
Table 6-7: Response Type Field Allowable Values from Responder	25
Table 6-8: Key for Table 6-7	25
Table 7-1: DISC_UNIQUE_BRANCH Response Packet Encoding	33
Table 7-2: DISC_UNIQUE_BRANCH Response Packet Decoding	34
Table 7-3: Control Field	34
Table 10-1: Required Response to Status Requests.....	51
Table 10-2: Date and Time Ranges	93
Table A-1: Command Class Defines.....	102
Table A-2: Response Type Defines	102
Table A-3: RDM Categories/Parameter ID Defines	103
Table A-4: Status Type Defines	104
Table A-5: Product Category Defines	105
Table A-6: Product Detail Defines.....	107
Table A-7: Preset Playback Defines	111
Table A-8: Lamp State Defines.....	111
Table A-9: Lamp On Mode Defines	111
Table A-10: Self Test Defines	112
Table A-11: Power State Defines.....	112
Table A-12: Sensor Type Defines.....	112
Table A-13: Sensor Unit Defines	114
Table A-14: Sensor Unit Prefix Defines	115
Table A-15: Data Type Defines.....	116
Table A-16 Parameter Description Command Class Defines	116
Table A-17: Response NACK Reason Code Defines	117
Table B-1: Status Message Markers.....	118
Table B-2: Status Message ID Definitions	118
Table C-1: Slot Type	120
Table C-2: Slot ID Definitions.....	120

List of Figures

Figure 2-1: Command Port Reference Circuit.....	6
Figure 4-2: Bit Asymmetrical Delay.....	13
Figure 7-1: Binary Search	30
Figure 7-2: Device Discovery Process.....	38
Figure F-1: Command Port Transmitter Test Circuit	129
Figure F-2: Command Port Load Test Circuit	130
Figure F-3: Calibration Circuit	131
Figure F-4: Responder Test Circuit.....	132

Introduction

The Remote Device Management Protocol (RDM) permits intelligent bi-directional communication between devices from multiple manufacturers utilizing a modified DMX512 data link. RDM is an EF 1.0 implementation of ANSI E1.11.

RDM permits a console or other controlling device to discover and then configure, monitor, and manage intermediate and end-devices connected through a DMX512 network. RDM provides for intelligent control of devices on a DMX512 network, which has not been previously available outside of proprietary networks.

This standard specifies: the physical layer and timings, device discovery process and algorithms, message structure and communication.

Overview

This document specifies the physical layer requirements for handling half-duplex bi-directional communication and the timings associated with bi-directional communication.

This document addresses requirements for controllers, end devices, and In-Line devices such as DMX512 splitters/mergers and distribution systems to implement or support RDM communication.

An RDM system functions as a polled system, meaning that no intermediate or end-device will initiate communication. Only the device acting as the controller shall have the capability to initiate a response from any RDM device.

The RDM protocol makes use of an Alternate START Code (ASC) as defined in ANSI E1.11, to establish communication on a conventional DMX512 link. The first step in the RDM process is for a controller to identify all the devices that are connected to the data link. This is accomplished by performing a binary-tree search, or other type of search, to identify the internal Unique ID (UID) of all the connected devices.

Once a device has been discovered, the controller can request status messages, or get and set device parameters such as the DMX512 Address. All messages sent are addressed to the UID of the targeted device or through one of the Broadcast UID addresses.

Devices that primarily act as controllers (and distribution devices that do their own "discovery") shall be referred to as controllers in this document. Devices that typically receive and act on DMX512 data and/or act on RDM messages shall be referred to as responding devices or responders. Controllers send requests and receive responses on their command port. Responders receive commands and send responses on their response port. An in-line device will typically have one response port to receive commands and NULL START CODE packets from the controller, and one or more command ports of their own to forward this data to their responders.

Only one controller can be active on a given DMX512 link at any one time.

During normal operation, it is expected that the Discovery and Parameter messages will be interspersed with normal NULL START Code DMX512 packets.

1 Normative References

ANSI E1.11-2004

*Entertainment Technology -- USITT DMX512-A --
Asynchronous Serial Digital Data Transmission Standard for
Controlling Lighting Equipment and Accessories.*

Entertainment Services and Technology Association (ESTA)
630 Ninth Avenue, Suite 609
New York, NY 10036
USA
1-212-244-1505
<http://tsp.esta.org>

ANSI/TIA/EIA-485-A-1998

*Electrical Characteristics of Generators & Receivers for Use in
Balanced Digital Multipoint Systems*

This standard will be referred to as EIA-485-A in this document.

Electronics Industries Alliance
2500 Wilson Boulevard
Arlington , VA 22201-3834 USA
1-703-907-7500
<http://www.eia.org/>

Telecommunications Industry Association
2500 Wilson Blvd., Suite 300
Arlington, VA 22201 USA
1-703- 907-7700 fax: 1-703-907-7727
<http://www.tiaonline.org/>

*Note: EIA-485-A is compatible with: ISO/IEC 8482:1993 Information Technology - Telecommunications and
information exchange between systems - Twisted pair multipoint interconnections.*

ISO/IEC 646

*Information Technology - ISO 7-bit Coded Character Set for
Information Interchange*

ISO 639-1

*Codes for the representation of names of languages –
Part 1: Alpha-2 code*

IEC
International Electrotechnical Commission
PO Box 131
3 rue de Varembe
1211 Geneva 20
Switzerland
41 22 919 02 11
www.iec.ch

ISO
International Organization for
Standardization
1, Rue de Varembe
Case Postale 56
CH-1211 Geneva 20
Switzerland
41 22 74 901 11
www.iso.ch

2 Physical Layer

2.1 General

E1.20 (RDM) is an extension to E1.11 (DMX512-A). A key goal of this standard is to allow the use of new and legacy DMX512 receiving devices in mixed systems with new E1.20 equipment and to provide a straightforward path to upgrade existing DMX512 distribution systems for support of the E1.20 protocol. The use of E1.20 devices in an E1.11 system will not compromise any E1.11 functionality.

The physical layer requirements of E1.20 are based on those of E1.11, with additional requirements related to bidirectional communication. Both developers and users must take note of these additional requirements in order to implement reliable E1.20 systems.

2.2 Electrical Specifications and Physical Layer Overview

The electrical specification of this Standard conforms to E1.11 (DMX512-A), except where specifically stated in this document. The E1.11 electrical specification is based on ANSI/TIA/EIA-485-A-1998. Where a conflict exists between E1.11 or ANSI/TIA/EIA-485-A-1998 and this document, this document is controlling as far as this standard is concerned.

2.2.1 ANSI E1.11 - EF1.0

Systems that comply with this standard fall within the scope of E1.11 Annex B - EF1. Systems that send data in both directions on the primary data link are classified as EF1. These systems shall use the primary data link for both the NULL START Code DMX512 packets and also return data controlled by the use of Alternate START Code packets.

Use of the secondary data link is beyond the scope of this standard.

2.3 Termination Requirements

One end of the primary data link shall be terminated as specified in E1.11.

The other end of the primary data link shall be terminated by a line biasing network. The requirements for line biasing networks are in Sections 2.4 and 2.5.

Command ports shall include the line biasing network.

Command ports may be designed with means to disconnect the line biasing network. In this configuration a line biasing network shall be provided by other means.

2.4 Maintaining data link state between packets

RDM systems use bidirectional half duplex operation. All command ports and responder ports are fitted with a transmitter and a receiver (e.g. a transceiver). Once a system is configured, only one transmitter is in the transmit mode at any one time. Typically, there may be considerable time when all transmitters are in a high impedance state. It is imperative that the data link shall maintain a marking state between packets, even if all transmitters are disabled.

To ensure this:

- 1.) A transmitter shall place the data link in a marking state at least 4 μ s before placing the line in a high impedance state.
- 2.) A transmitter shall begin driving the line in a marking condition.
- 3.) A Command port located at the end of a data link shall employ a line bias network as detailed in Section 2.4.1.
- 4.) A Command port not located at an end of the data link shall not employ the line bias network of Section 2.4.1. Means of termination and biasing of such systems is beyond the scope of this standard.

2.4.1 Line Bias Networks

The command port shall provide a means to bias the termination of the data link to a value of at least 245 mV and verified by using the test circuit described in Appendix F. This means may be disabled for special applications. If the line biasing network is enabled, the differential input impedance shall be 120ohms +/- 10%; however, if it is not enabled, the differential input impedance shall not be greater than one unit load.

The termination shall be polarized such that Data+ of the data link is positive with respect to Data- of the data link. The Line Biasing network shall maintain this bias when the data link is loaded with the equivalent of 32 unit loads and common mode voltage is varied over the range of +7 volts to -7 volts DC.

2.5 Command port reference circuit

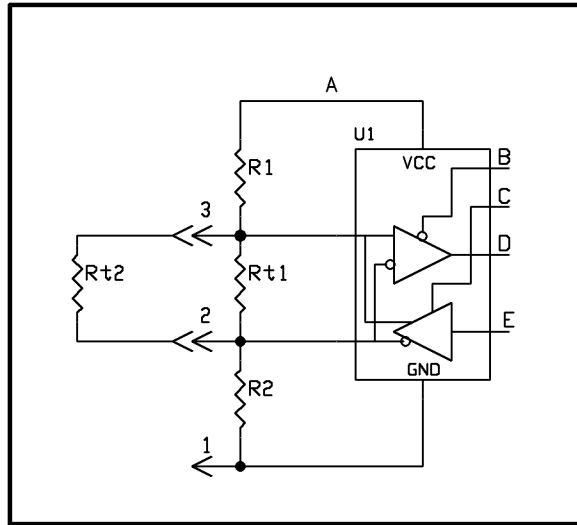


Figure 2-1: Command Port Reference Circuit

This standard does not mandate a specific bias circuit. Figure 2-1 is provided as a reference.

Key for Figure 2-1:

- 1) Data link common (0 Volt DC)
- 2) Data-
- 3) Data+
- A) +5 Volt DC supply
- U1) EIA485 transceiver
- B, C) Transceiver control lines
- D, E) Logic level data lines

2.5.1 Details of the reference circuit.

The reference line bias network is a simple voltage divider consisting of resistors R1, Rt1 in parallel with Rt2, and R2. Rt1 is the termination at the command port. Rt2 is the terminating resistor at the opposite end of the data link. The network is connected between the Vcc power supply and supply common. The connection between R1 and Rt1 is connected to the Data+ output of the transceiver. The connection between Rt1 and R2 is connected to the Data- output of the transceiver.

The physical line bias network consists of R1, Rt1, and R2. Correct operation is dependent upon the presence of Rt2. These component values are specified in Table 2-1.

Table 2-1: Command Port Reference Circuit Values

R1	562 Ω 2%	
R2	562 Ω 2%	
Rt1	133 Ω 2%	Rt1 parallel (R1+R2) should equal 120 Ω
Rt2	120 Ω 5%	This resistor is not physically part of the line bias network. It should be located at the opposite end of the data link from the command port.
Vcc	+5 VDC	Resistor values above assume 5VDC operation. Other Vcc values will require new resistor value calculations.

2.6 Default line state control

It is recommended that Responders and In-Line devices prevent their responder ports from erroneously driving the line in case of failure.

All ports shall power up with the driver in a high impedance state.

The Line Bias network shall be enabled prior to RDM communication beginning.

3 Timing

RDM systems can carry several different types of packets. Controllers can generate NULL START Code packets, RDM Alternate START Code (ASC) packets and non-RDM ASC packets. RDM packets are compatible with DMX512 and DMX512-A timing. Certain timing requirements have been tightened to ensure correct operation of the RDM protocol.

RDM controllers generate RDM request messages that are identified by the RDM ASC. Responders can generate an RDM response, also identified by the RDM ASC, but only immediately after receiving a controller request. In general, every controller request will generate a responder response. However, there will be times, either for protocol reasons or through errors, that a controller request may not cause the generation of a device response.

Because RDM is a half duplex communications protocol, the controller must release (stop driving) the line before the responders begin to drive the line. To ensure that this “line turn around” takes place properly, a certain amount of dead time has been allocated to this function. This time is relatively large to allow microcontrollers/microprocessors to disable the driver after the last byte has been sent and to allow for in-line devices that may cause timing delay.

The DISC_UNIQUE_BRANCH response message is an exception to normal timing rules, as it does not contain a BREAK or MAB. See Section 7.5 Discovery Unique Branch Message for additional details on this message.

3.1 Controller Timing Requirements

3.1.1 Controller Packet Timing

RDM controller equipment shall conform to the timing in Table 3-1. The packet structure showing the various timing elements is represented by the Timing Diagram (Figure 5) located in E1.11.

Table 3-1: Controller Packet Timing

		Break		MAB		Inter-slot Time		Total Packet Time
		Min	Max	Min	Max	Min	Max	Max
1	Transmit	176µs	352µs	12µs	88µs	0µs	2.0ms ^A	440µs ^B + (n ^C x 44µs ^D) + ((n ^C - 1) x 76µs ^E)
2	Receive	88µs	352µs	8µs	88µs	0µs	2.1ms	

Notes:

- A). This is the maximum inter-slot time for any individual slot, However, the Total Packet Time requirement must be observed.
- B). The 440µs in the total packet time formula is the sum of the maximum break time and maximum MAB.
- C). n is the number of data slots in the packet.
- D). Slot Time per ANSI E1.11.
- E). The average inter-slot time shall not exceed 76µs.

All Controller generated packets, regardless of Start Code, shall conform to line one of Table 3-1. Controllers shall correctly receive packets that conform to line two of Table 3-1.

Break timing has been modified from the DMX512 standard. The minimum has been lengthened to allow In-Line devices to decrease this time as they pass the data through. The maximum break time has been set to ensure RDM command throughput and to ensure that RDM has minimal impact on the DMX512 update rate. To this end, it is recommended that the break, MAB, and inter-slot times be kept as close to the minimum as possible.

Controllers may consider responder packets with inter-slot delays exceeding the maximum defined in line 2 of Table 3-1 to be lost and may resume sending controller packets.

3.1.2 Controller Packet spacing

Correct RDM performance requires certain minimum and maximum packet spacing.

The start of a packet (Start of Packet or SOP) is defined as the leading edge of the BREAK.

The end of a packet (End of Packet or EOP) is defined as the end of the second stop bit of the last slot. All times are shown from the EOP to the SOP at the controller.

RDM controllers shall conform to the timing specified in Table 3-2 for all packet timing. These times are specified at the controller port.

Table 3-2: Controller Packet Spacing Times

Line	Description	Message Sequence	Line Turn Around?	Min ¹	Max
1	Discovery Response	Controller: Discovery -> Responder: Response	Yes	176 μ s	2.8 ms ²
2	Discovery	Controller: Discovery -> Controller: Any Packet	Yes	5.8 ms ^{2,3}	1 s ⁴
3	Normal Operation	Controller: Request ⁵ -> Responder: Response	Yes	176 μ s	2.8 ms ²
4	Normal Operation	Responder: Response -> Controller: Any Packet	Yes	176 μ s	1 s ⁴
5	Missing response	Controller: Request -> Controller: Any Packet ⁶	Yes, response lost	3 ms ²	1 s
6	Normal, No Response Expected	Controller: Broadcast ⁷ -> Controller: Any Packet ⁸	No	176 μ s	1 s ⁴
7	Normal, No Response Expected	Controller: Non-RDM -> Controller: Any Packet	No	176 μ s	1 s ⁴

Table 3-2 Notes:

- 1.) Minimum time is to allow the data link to turn around.
- 2.) These times include 704 μ s of system delay time. See Section 4.2.2 Data Delay.
- 3.) $(44\mu\text{s} \times 24 \text{ bytes}) + (76\mu\text{s} \times 23) = 2.804\text{ms}$ packet time rounded to 2.9ms. 2ms response time + 2.9ms packet time + 0.7ms of system delay time + 0.2ms to assure that any in-line device has turned around = 5.8ms. Note that discovery responses are sent without breaks. See Section 7.5 Discovery Unique Branch Message.
- 4.) This time should be kept short if high update rates are required.
- 5.) A Request is any controller message to which a response is expected.
- 6.) Missed Response: This covers the case of a missing response. If the response is received, then Table 3-2, Line 3 takes precedence. This includes the 704 μ s of system delay plus an additional 300 μ s of delay to allow for a late responder.
- 7.) A non-discovery Broadcast is a controller message to which no response is allowed.
- 8.) Any Packet can be a DMX512 NULL START Code, an RDM ASC or a non-RDM ASC packet.

3.1.3 Driver shutoff time

When a response is expected, the controller shall place its driver in high impedance state no later than 88µs after the end of the last stop bit of the last slot of that message. The controller should continue to drive the line if the next packet is known to be generated by the controller. Controller drivers shall continue to drive a MARK on the line after the last stop bit and until the driver is disabled. Controller drivers shall drive the line (not enter a high impedance state) throughout the entire packet.

3.2 Responder Timing Requirements

3.2.1 Responder Packet Timings

RDM responders shall conform to the timing specified in Table 3-3. The packet structure showing the various timing elements is represented by the DMX512-A Timing Diagram (Figure 5) located in ANSI E1.11.

Table 3-3: Responder Packet Timing

		Break		MAB		Inter-slot Time		Total Packet Time
		Min	Max	Min	Max	Min	Max	Max
1	Receive	88µs	1s	8µs	1s	0µs	2.1ms	
2	Transmit	176µs	352µs	12µs	88µs	0µs	2.0ms ^A	$440\mu s^B + (n^C \times 44\mu s^D) + ((n^C - 1) \times 76\mu s^E)$
3	Transmit Discovery Response	N/A	N/A	N/A	N/A	0µs	2.0ms ^A	2.9ms

Notes:

- A). This is the maximum inter-slot time for any individual slot, However, the Total Packet Time requirement must be observed.
- B). The 440µs in the total packet time formula is the sum of the maximum break time and maximum MAB.
- C). n is the number of data slots in the packet.
- D). Slot Time per ANSI E1.11.
- E). The average inter-slot time shall not exceed 76µs.

All Responder generated non-Discovery packets shall conform to line two of Table 3-3. Responders shall correctly receive packets that conform to line one of Table 3-3.

Break timing has been modified from the DMX512 standard. The minimum has been lengthened to allow In-Line devices to decrease this time as they pass the data through. The maximum break time has been set to ensure RDM command throughput and so that RDM has minimal impact on the DMX512 update rate. To this end, it is recommended that the break, MAB, and inter-slot times be kept as close to the minimum as possible.

Responders may consider controller packets with inter-slot delays exceeding the maximum in line 1 of Table 3-3 to be lost.

3.2.2 Responder Packet spacing

Correct RDM performance requires certain minimum and maximum packet spacing.

The start of a packet (Start of Packet or SOP) is defined as the leading edge of the BREAK. In the case of break-less responses (DISC_UNIQUE_BRANCH), the SOP is defined as the leading edge of the start bit of the first slot sent by the Responder.

The end of packet (End of Packet or EOP) is defined as the trailing edge of the second stop bit of the last slot transmitted. The "trailing edge" is defined as 44µs following the leading edge of the start bit. Note that there is no physical transition at the "trailing edge" of the stop bit as the line must be left in a Marking state.

All times are shown in Table 3-4 are from the Controller's EOP to the Responder's SOP at the responder.

RDM responders shall conform to the timings specified in Table 3-4.

Table 3-4: Responder Packet Spacing Times

Line	Description	Min ¹	Max
1	Controller: Request ² -> Responder: Response	176µs	2ms
2	Controller: Discovery -> Responder: Response	176µs	2ms

Table 3-4 Notes:

- 1.) Minimum time is to allow the data link to turn around.
- 2.) A Request is any controller message to which a response is expected.

3.2.3 Responder discovery response driver enable time

Responders shall not drive the line in a marking state for more than 12µs prior to the beginning of the first start bit.

3.2.4 Driver shutoff time

All responders shall place their drivers in a high impedance state no later than 88µs after the end of the last stop bit of the last slot of a response. The responder's driver shall continue to drive a MARK on the line after the last stop bit and until the driver is disabled. The responder's driver shall drive the line (not enter a high impedance state) throughout the entire packet.

3.2.5 Discovery Response MARK time

Responders may wish to drive a MARK on the line for at least 4µs prior to the first start bit of the discovery response. This will ensure that all other devices on the line recognize the start bit.

3.3 Data collisions

A correctly functioning RDM system operates without data collisions, except during Discovery (Section 7). During device discovery there will frequently be data collisions. While many collisions may be detected at the protocol level, some collisions could be interpreted as properly framed packets. While the physical layer does not impose any special requirements for hardware detection of collisions, it is recommended that proper character framing is verified by both controllers and responders.

4 In-Line Devices

DMX512 systems use several types of In-Line devices (such as a splitter or merger) to distribute the data. Since RDM uses bi-directional communication, RDM In-Line devices have additional operational requirements compared to non-RDM In-Line devices. This section describes requirements unique to RDM In-Line devices.

4.1 Overview

There are two ways in-line devices can operate: Transparent and Proxy. This section pertains to Transparent In-line devices. Proxy devices are described in Section 8.

On In-Line devices, the port receiving data communication from the Controller shall be designated as the Responder Port. The ports generating data communication to the end-devices shall be designated as the Command Port(s). For example, a splitter would typically have a single Responder Port with multiple Command Ports.

4.1.1 Transparent In-Line devices

A Transparent In-line device provides bi-directional transfer of data between the response and command ports. It does not initiate discovery on its command ports.

Transparent In-line devices may make use of the packet data (e.g. Slot Count) to maintain proper timings. Such devices may generate their own packet/slot timing but should have minimal impact on packet spacing. A Transparent In-line device may or may not be discoverable as an RDM device.

4.2 Transparent In-Line Device Timing Requirements

Transparent In-Line devices of different types may be cascaded to create the RDM distribution network. Such devices must abide by certain timing restrictions to allow a system to operate within the requirements outlined in Section 3.2 Responder Timing Requirements.

4.2.1 Port Turnaround

After receiving an RDM request packet, the in-line device shall switch to receiving data at its Command Ports, within 132 μ s of the end of the RDM packet.

After receiving an RDM request packet, the first port that is pulled to a low state for the start of a BREAK becomes the active port. Note that this port may be the responder port, in which case the in-line device shall return to forward data flow. Otherwise, data from the active port shall drive the responder port and may drive all other command ports on the in-line device.

After the EOP of a response packet an in-line device shall be capable of receiving data on its responder port (and returning to forward data flow) within 132 μ s.

4.2.1.1 Port Turnaround during Discovery

Because there may not be a recognizable EOP for the response, the in-line device shall return to forward data flow no sooner than Table 3-2 Line 2 Minimum Time minus 200 μ s from Table 3-2 Note 3. The in-line device shall be capable of returning to forward data flow no later than Table 3-2 Line 2 Minimum Time.

4.2.2 Data Delay

Transparent In-Line devices shall not delay the data by more than 88 μ s. This is a maximum delay. Delay times can be shorter, theoretically 0 μ s. This allows operation with up to four Transparent In-Line devices between the controller and responder. This accounts for the 704 μ s (352 μ s send and 352 μ s receive) of system delay time included in Section 3.1.2 Controller Packet spacing

4.2.3 Bit Distortion

Some In-Line devices distort bit times due, in part, to unsymmetrical propagation delays and transition times (the low to high time differs from the high to low time). Total bit time distortion shall be limited to a maximum of +/- 1.875% (75ns). This delay is not cumulative during a slot.

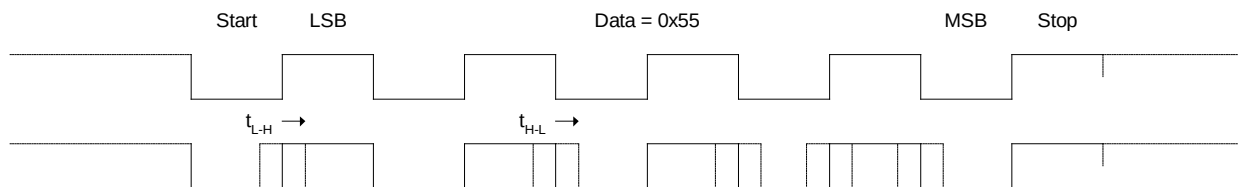


Figure 4-2: Bit Asymmetrical Delay

Transparent In-Line devices may regenerate the slot timings. If so, they shall conform to the slot timing requirements in Section 3.

4.2.4 BREAK timing

RDM timing allows up to four cascaded Transparent In-Line devices.

In-line devices shall be permitted to shorten the duration of the BREAK immediately following an RDM request packet. The BREAK shall not be shortened by more than 22 μ s. This requirement allows for systems with four In-line devices in series, while assuring that breaks are always 88 μ s or longer.

Manufacturers of in-line DMX512 processing or distribution devices shall declare any shortening of the BREAK.

4.3 In-Line devices operating as part of Non-RDM system

RDM In-Line Devices shall conform to the packet timing requirements of E1.11 when used in a non-RDM system, including the correct reception and retransmission of NSC packets with the minimum 88 μ s break.

4.4 In-line Devices as Responders

An in-line device may also be a responder. This allows for status monitoring or configuration of the in-line device itself.

5 Device Addressing

5.1 General

Responders and Controllers identify themselves with a 48-bit Unique ID (UID). The Unique ID (UID) consists of a 16-bit ESTA assigned Manufacturer ID with a 32-bit Device ID, as shown in Table 5-1.

Table 5-1: Unique ID (UID) Format

ESTA Manufacturer ID (16-bit)	Device ID (32-bit)
----------------------------------	-----------------------

The 32-bit device ID shall be unique throughout all products manufactured under a specific Manufacturer ID, to ensure that no two devices with the same UID will appear on the data link.

For use in this standard the value of the 16-bit Manufacturer ID shall be restricted to 0x0001 through 0x7FFF.

A responder that is not acting as a proxy shall only respond to messages addressed to its UID.

A responder that acts a proxy device shall only respond to messages addressed to its own UID, or the UID of one of the represented devices.

In general, each Responder port should be assigned a single Device ID. The Sub-Device mechanism (see Section 9) is provided to support the logical organization of a device's components.

Multiple Command ports on a device that originate messages may use the same Device ID, but shall identify each port uniquely using the Port ID/Response Type field as detailed in Section 6.2.7.

Information on Manufacturer ID Registration can be found in E1.11 Annex E.

5.2 UID Text Representation

The recommended method for representing the UID in text is in hexadecimal format with a colon separating the Manufacturer ID and the Device ID.

An example of such would be: mmmm:dddddddd, where mmmm is the Manufacturer ID in hexadecimal and dddddddd is the Device ID in hexadecimal.

5.3 Broadcast Message Addressing

A message may be addressed to multiple responders by using a special value in the Destination UID field. This technique is commonly used with Discovery Messages and may also be used with any SET_COMMAND message. Messages may be addressed to all devices, or to all devices of a specific manufacturer.

To address a message to all devices regardless of Manufacturer, the BROADCAST_ALL_DEVICES_ID shall be used in the Destination UID field.

To address a message to all devices of a specific Manufacturer, the ALL_DEVICES_ID shall be used with the desired Manufacturer ID in the Destination UID field.

When Broadcast Addressing is used for non-Discovery messages, the responders shall not send a response.

5.4 Responders with multiple Responder Ports in a single Device.

Some devices may have multiple responder ports that are Discoverable (i.e., they respond to DISC_UNIQUE_BRANCH messages). On these devices, each responder port shall identify itself with a unique Device ID.

Furthermore, one responder port shall be designated as the primary port, to which the other responder ports are bound, allowing the responder ports to be associated together as a single physical device. The implementation of Binding Device ID's is discussed in Section 7.6.2.

6 Message Structure

All RDM packets shall use the following message structure, with the exception of the Discovery Unique Branch response message. The format of the Discovery Unique Branch message is detailed in Section 7.5 Discovery Unique Branch Message.

6.1 Byte Ordering

All multi-byte data shall be transmitted in Big-Endian order.

For example, 0x01AB02CD would be transmitted as shown in Table 6-1.

Table 6-1: Byte Ordering

Slot #	Data	
i	0x01	Most Significant Byte
i+1	0xAB	
i+2	0x02	
i+3	0xCD	Least Significant Byte

6.2 Packet Format

All fields within the packet shown in Table 6-2 are assumed to be 8-bit values unless otherwise stated.

Table 6-2: Packet Format

START Code		
Sub-Start Code		
Message Length		
Destination UID (48-bit)		
Source UID (48-bit)		
Transaction Number (TN)		
Port ID / Response Type		
Message Count		
Sub-Device (16-bit)		
Message Data Block (MDB) (Variable Size)		
Checksum (16-bit)		

6.2.1 START Code

This field shall contain the defined RDM START Code (SC_RDM). Controllers and Responders shall always send SC_RDM in this slot, and any packet containing a value other than SC_RDM is outside the scope of this standard.

6.2.2 Sub START Code

This field shall contain the Sub-START Code within RDM that defines this packet structure (SC_SUB_MESSAGE). Future versions of this standard which may have additional or different packet structures would use this field to identify the packet structure being used.

Controllers shall always send SC_SUB_MESSAGE in this slot, and Responders shall ignore any packets containing other values.

6.2.3 Message Length

The Message Length value is defined as the number of slots in the RDM Packet including the START Code and excluding the Checksum. Each slot is an 8-bit value.

The Message Length field points to the Checksum High Slot.

Table 6-3 below illustrates the relationship of Message Length to the Checksum using the Get Status Messages command as an example.

Table 6-3: Example of Packet showing Message Length Pointer to Checksum

Slot #	Description	Data Value	Remarks
0	START Code	SC_RDM	
1	Sub-START Code	SC_SUB_MESSAGE	
2	Message Length	25 (Slot # of Checksum High)	Range 24 to 255.
3 – 8	Destination UID	0x123456789ABC	
9 – 14	Source UID	0xCBA987654321	
15	Transaction Number (TN)	0	
16	Port ID / Response Type	1	
17	Message Count	0	
18 - 19	Sub-Device	0	
20	Command Class (CC)	GET_COMMAND	
21 – 22	Parameter ID (PID)	STATUS_MESSAGES	
23	Parameter Data Length (PDL)	1	Range 0 to 231
24	Parameter Data (PD)	STATUS_ERROR	
25	Checksum High		
26	Checksum Low		

6.2.4 Destination UID

The Destination UID is the UID of the target device(s).

6.2.5 Source UID

The Source UID is the UID of the device originating this packet.

6.2.6 Transaction Number (TN)

The Transaction Number is an unsigned 8-bit field. Controller generated packets increment this field every time an RDM packet is transmitted. This field shall be initialized to 0 and roll over from 255 to 0.

Responders shall reply with their Transaction Number set to the Transaction Number contained in the controller packet to which they are responding.

The Transaction Number can be used to help match a response message to the Controller's request.

6.2.7 Port ID / Response Type

This field serves different functions depending on whether the message is being generated by the controller or the responder.

6.2.7.1 Port ID / Response Type field for Controller Generated Messages

For Controller generated messages (GET_COMMAND, SET_COMMAND, and DISCOVERY_COMMAND), the Port ID field shall be set in the range of 1-255 identifying the Controller Port being used, such that the combination of Source UID and Port ID will uniquely identify the controller and port where the message originated.

6.2.7.2 Port ID / Response Type field for Responder Generated Messages

For Responder generated messages (GET_COMMAND_RESPONSE, SET_COMMAND_RESPONSE, and DISCOVERY_COMMAND_RESPONSE), this field is used as the Response Type field. Response Type Field usage is detailed in Section 6.3.

6.2.8 Message Count

The message count field is used by a responder to indicate that additional data is now available for collection by a controller. This data (which might be unrelated to the current message transaction) should be collected by the controller using the GET:QUEUED_MESSAGE command.

6.2.8.1 Message Count field for Controller Generated Messages

The Message Count shall be set to 0x00 in all controller generated requests.

6.2.8.2 Message Count field for Responder Generated Messages

The Message Count shall be incremented by a responder whenever there is a new message pending collection by a controller. Thus a controller can determine, from any response, the number of queued messages pending.

When replying to a GET: QUEUED_MESSAGE command, a responder shall decrement the Message Count prior to responding, unless it is already zero.

If a responder has more than 255 messages queued, then the Message Count field shall remain at 255 until the number of queued messages is reduced below that number.

The following sequence of messages illustrates the use of the Message Count field.

Controller sends GET: LANGUAGE

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

Responder replies with ACK and the current language. Message Count shows 0x02 indicating that two other responses are ready for retrieval.

(Response Type) ACK	(Message Count) 0x02	(Sub-Device) 0x0000
(CC) GET_COMMAND_RESPONSE	(PID) LANGUAGE	(PDL) 0x02
(PD)		
2 character alpha code for ISO 639-1		

Controller sends GET: QUEUED_MESSAGE:

(Port ID) 0x01 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000
(CC) GET_COMMAND	(PID) QUEUED_MESSAGE	(PDL) 0x01
(PD)		
STATUS_ERROR		

Responder sends queued message with Message Count decremented showing only 1 pending messages remains.

(Response Type) ACK	(Message Count) 0x01	(Sub-Device) 0x0000
(CC) GET_COMMAND_RESPONSE	(PID) DMX_START_ADDRESS	(PDL) 0x02
(PD)		
DMX512 Address (16-bit)		

Controller sends another GET: QUEUED_MESSAGE:

(Port ID) 0x01 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000
(CC) GET_COMMAND	(PID) QUEUED_MESSAGE	(PDL) 0x01
(PD)		
STATUS_ERROR		

Responder sends queued message with Message Count decremented (and now zero) showing no further pending messages exist.

(Response Type) ACK	(Message Count) 0x00	(Sub-Device) 0x0000
(CC) GET_COMMAND_RESPONSE	(PID) DMX_PERSONALITY	(PDL) 0x02
(PD)		
Current Personality # of Personalities		

6.2.9 Sub-Device Field

Sub-devices should be used in devices containing a repetitive number of similar modules, such as a dimmer rack. The Sub-Device field allows Parameter messages to be addressed to a specific module within the device to set or get properties of that module.

The 16-bit sub-device field provides a range of 512 valid sub-devices, addressed from 1 - 512. The value of 0xFFFF is reserved as a SUB_DEVICE_ALL_CALL. A value of 0x0000 shall be used to address the root or base properties of the device that do not belong to any sub-device module.

The Parameter ID designates which parameter on the sub-device is being addressed.

The use of Sub-Devices is described further in Section 9.

6.2.10 Message Data Block (MDB)

Table 6-4 shows the format of the Message Data Block portion of the data packet from Table 6-2.

Table 6-4: Message Data Block (MDB) Format

Command Class (CC)	Parameter ID (PID) [16-bit]	Parameter Data Length (PDL)	Parameter Data (PD) [Variable Length]
-----------------------	--------------------------------	--------------------------------	--

6.2.10.1 Command Class (CC)

The Command Class (CC) in Table 6-5 specifies the action of the message.

Table 6-5: Command Class Description

Command Class	Response Expected	Description
GET_COMMAND	YES	Request the value or status of a parameter from the device.
GET_COMMAND_RESPONSE		Response from a GET_COMMAND message.
SET_COMMAND	YES	Change the value of a parameter within the device.
SET_COMMAND_RESPONSE		Response from a SET_COMMAND message.
DISCOVERY_COMMAND	YES	Message related to the Discovery Process.
DISCOVERY_COMMAND_RESPONSE		Response from a DISCOVERY_COMMAND Message.

Responders shall always generate a response to GET_COMMAND and SET_COMMAND messages except when the Destination UID of the message is a broadcast address. Responders shall not respond to commands sent using Broadcast addressing, in order to prevent collisions.

Command Class values are enumerated in Table A-1.

6.2.10.2 Parameter ID (PID)

The Parameter ID is a 16-bit number that identifies a specific type of Parameter Data.

The Parameter ID (PID) may represent either a well known Parameter such as those defined in this document, or a Manufacturer-specific parameter whose details are either published by the Manufacturer for third-party support or proprietary for the Manufacturer's own use.

ESTA defined PID's shall be in the range of 0x0000 – 0x7FDF and are enumerated in Table A-3. PID's in the range of 0x7FE0 – 0x7FFF are reserved for future uses of this standard. PID's in Table A-3 are organized into categories to create logical groupings.

Manufacturer-specific PID's shall be created in the range of 0x8000 – 0xFFDF. Uniqueness of PID's in this range is accomplished by associating the PID with the Manufacturer ID found as the most significant 16-bits of the UID. PID's in the range of 0xFFE0 – 0xFFFF are reserved for future uses of this standard.

Manufacturer-Specific PID values should be selected by choosing the appropriate category from Table A-3 and adding an offset of 0x8000 to preserve a logical organization.

A manufacturer shall not use the same manufacturer-specific PID value for more than one meaning within any products falling under a given Manufacturer ID.

Controllers shall not send Manufacturer-specific PID messages addressed to the all manufacturer BROADCAST_ALL_DEVICES_ID. Responders shall ignore any manufacturer-specific message addressed to the BROADCAST_ALL_DEVICES_ID.

6.2.10.2.1 Minimum Required Parameter Support

All devices shall support the minimum set of PID's indicated by the "Required" column of Table A-3.

6.2.10.3 Parameter Data Length (PDL)

The Parameter Data Length (PDL) is the number of slots included in the Parameter Data area that it precedes. When this field is set to 0x00 it indicates that there is no Parameter Data following.

6.2.10.4 Parameter Data (PD)

The Parameter Data is of variable length. The content format is PID dependent.

6.2.11 Checksum

The Checksum field is the unsigned, modulo 0x10000, 16-bit additive checksum of the entire packet's slot data, including START Code. The checksum is an additive sum of the 8-bit fields into a 16-bit response value.

If the checksum field in the packet does not match the calculated checksum, then the packet shall be discarded and no response sent.

Table 6-6 shows an example of how the checksum is calculated. In the example below, the Checksum is the sum of all the slots from Slot 0 to Slot 24.

Table 6-6: Checksum Usage Example

Slot #	Description	Data Value	Remarks
0	START Code	0xCC	SC_RDM
1	Sub START Code	0x01	SC_SUB_MESSAGE
2	Message Length	0x19	Slot # of Checksum High = 25
3	Destination UID	0x12	0x123456789abc
4		0x34	
5		0x56	
6		0x78	
7		0x9A	
8		0xBC	
9	Source UID	0xCB	0xcba987654321
10		0xA9	
11		0x87	
12		0x65	
13		0x43	
14		0x21	
15	Transaction Number	0x00	
16	Port ID / Response Type	0x01	
17	Message Count	0x00	
18	Sub-Device	0x00	Root Device
19		0x00	
20	Command Class	0x20	GET_COMMAND
21	Parameter ID	0x00	STATUS_MESSAGES
22		0x30	
23	Parameter Data Length	0x01	
24	Parameter Data	0x04	STATUS_ERROR
25	Checksum High	0x06	0x066A
26	Checksum Low	0x6A	

6.3 Response Type Field Values

The Response Type field is used in messages from Responders to indicate the acknowledgement type of the response.

Response Type values are enumerated in Table A-2. Table 6-7 below indicates the valid codes for the Response Type field entry. This table does not indicate whether a response occurs, only the allowed values of the Response Type field if a response occurs.

Table 6-7: Response Type Field Allowable Values from Responder

Command Class	ACK	ACK_ OVERFLOW	ACK_ TIMER	NACK_ REASON
DISCOVERY_ COMMAND_ RESPONSE	✓	X	X	X
GET_COMMAND_ RESPONSE	✓	✓	✓	✓
SET_COMMAND_ RESPONSE	✓	✓	✓	✓

Table 6-8: Key for Table 6-7

Icon	Notes
✓	Response Type Allowed
X	Response Type Not Allowed

6.3.1 Acknowledge (RESPONSE_TYPE_ACK)

The response RESPONSE_TYPE_ACK indicates that the responder has correctly received the controller message and is acting upon the message.

The format of the MDB of the message is defined in the corresponding message definition.

6.3.2 Acknowledge Overflow (RESPONSE_TYPE_ACK_OVERFLOW)

The response RESPONSE_TYPE_ACK_OVERFLOW indicates that the responder has correctly received the controller message and is acting upon the message, but there is more response data available than will fit in a single response message.

This message shall be used in cases such as GET: PROXIED_DEVICES and GET: SUPPORTED_PARAMETERS where the response data is larger than can fit in a single response message.

To receive the remaining data, the controller should continue to send GET_COMMANDS for the same PID. The responder shall send subsequent blocks of data with a response type of RESPONSE_TYPE_ACK_OVERFLOW until the remaining data can fit in a single message. The responder shall set the response type to RESPONSE_TYPE_ACK on the final response message in the sequence, to indicate completion of the data transfer.

The responder shall not queue up subsequent messages when a GET_COMMAND results in a response type of RESPONSE_TYPE_ACK_OVERFLOW. This allows the controller to explicitly manage the transfer of larger data blocks.

The responder shall abort a partial transfer of overflow data for a PID when receiving a command for a different PID before the overflow data transfer is complete. A subsequent command for the overflow PID will result in a new data transfer starting at the beginning of the data set.

The format of the MDB of the message is defined in the corresponding message definition.

An example of this Response Type is listed below:

Controller sends GET: PROXIED_DEVICES

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

Responder sends response message indicating overflow condition, that more data is available.

(Response Type) ACK_OVERFLOW	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000
(CC) GET_COMMAND_RESPONSE	(PID) PROXIED_DEVICES	(PDL) 0xE4
(PD) Packed Field with 38 UID's (48-bits each). Maximum number of UID's that can be fit within PDL limit.		

Controller sends another GET: PROXIED_DEVICES

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

Responder sends final response message in sequence:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000
(CC) GET_COMMAND_ RESPONSE	(PID) PROXIED_DEVICES	(PDL) 0x1E
(PD) Packed Field with 5 UID's (48-bits each).		

6.3.3 Acknowledge Timer (RESPONSE_TYPE_ACK_TIMER)

RESPONSE_TYPE_ACK_TIMER indicates that the responder is unable to supply the requested GET information or SET confirmation within the required response time.

The format of the MDB of the response message is defined as follows:

Response:

(Response Type) ACK_TIMER	(Message Count) OldMC	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) Copy of Controller PID	(PDL) 0x02
(PD) Estimated Response Time (16-bit)		

The Estimated Response Time is a 16-bit unsigned field that defines the estimated time in tenths of a second (100ms increments) that will elapse before the responder can provide the required information.

When the responder is able to provide the required information, it shall add the correctly formatted GET_COMMAND_RESPONSE message or SET_COMMAND_RESPONSE message to its QUEUED_MESSAGE list. It shall also increment its Message Count at the time the message is added to the QUEUED_MESSAGE list.

This mechanism allows the controller to retrieve deferred Get command data by issuing GET: QUEUED_MESSAGE.

The following transaction provides an example of this mechanism:

Controller sends GET: LAMP_STRIKES

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

Responder is unable to retrieve data and replies with ACK_TIMER indicating it will need approximately 60 seconds to process the request:

(Response Type) ACK_TIMER	(Message Count) 0x00	(Sub-Device) 0x0000
(CC) GET_COMMAND_RESPONSE	(PID) LAMP_STRIKES	(PDL) 0x02
(PD)		
0x0258		

Responder data becomes available for response and has been added to the QUEUED_MESSAGE list. Any intervening messages would show an incremented message count.

80 seconds later

Controller sends GET: QUEUED_MESSAGE:

(Port ID) 0x01	(Message Count) 0x00	(Sub-Device) 0x0000
(CC) GET_COMMAND	(PID) QUEUED_MESSAGE	(PDL) 0x01
(PD)		
STATUS_ERROR		

Responder sends queued message:

(Response Type) ACK	(Message Count) 0x00	(Sub-Device) 0x0000
(CC) GET_COMMAND_RESPONSE	(PID) LAMP_STRIKES	(PDL) 0x04
(PD)		
Lamp Strikes (32-bit)		

6.3.4 Negative Acknowledge (RESPONSE_TYPE_NACK_REASON)

The response RESPONSE_TYPE_NACK_REASON shall only be used in conjunction with the Command Classes GET_COMMAND_RESPONSE & SET_COMMAND_RESPONSE.

RESPONSE_TYPE_NACK_REASON indicates that the responder is unable to reply with the requested GET information or unable to process the specified SET command.

The format of the MDB of the response message is defined as follows:

Response:

(Response Type) NACK_REASON	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_RESPONSE	(PID) Copy of Controller PID	(PDL) 0x02	
(PD)			
NACK Reason Code (16-bit)			

The NACK Reason code defines the reason that the responder is unable to comply with the request.

Valid NACK Reason Codes are enumerated in Table A-17.

7 Discovery Method

7.1 General

Responders may be discovered by conducting a binary search based on the 48-bit UID. Collisions are expected as part of the discovery process. Once all responders have been discovered, the system operates in a collision-free environment by utilizing the UID's for addressing messages. While search methods other than a binary search are allowed, this is the preferred method and is described in this section.

7.2 Binary Search Tree

Figure 7-1 illustrates a Binary Search Tree. Each node of the tree represents a decision fork for the search, and is labeled with a value pair representing the corresponding starting and ending UID values for that branch of the search.

The controller discovers each device by working through the tree from right to left exploring each branch that provides a response.

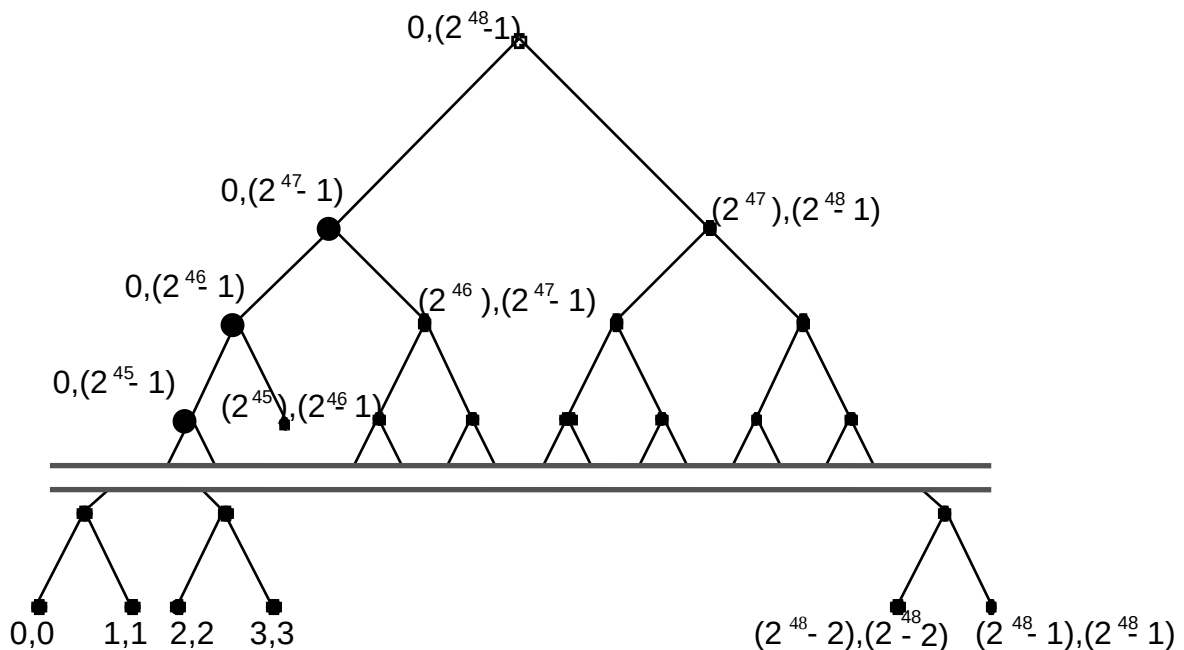


Figure 7-1: Binary Search

7.3 Discovery Process Steps

Devices respond to discovery messages (DISC_UNIQUE_BRANCH) unless they have been muted for discovery. The normal discovery process consists of un-muting all devices, and as they are individually discovered, muting each device. The devices must be muted as they are discovered so that they will not respond and create response collisions as the controller attempts to discover other devices. In general, the discovery process proceeds as follows:

1. For full Discovery, clear all Discovery Mute flags by sending the DISC_UN_MUTE message with the Destination UID set to BROADCAST_ALL_DEVICES_ID.
2. Send a DISC_UNIQUE_BRANCH message with parameter data corresponding to the current branch of the binary search tree. Any device with a UID greater than or equal to the Lower Bound and less than or equal to the Upper Bound that was transmitted in the Parameter Data area will provide a response message.
3. Numerous collisions are expected. Any response indicates devices within that UID range. No response indicates there are no devices within that branch.
4. If the checksum is valid for the response, attempt to Mute the device at the UID included in the response by sending the DISC_MUTE message. If a device is located at that UID then it will provide a response message and will no longer respond to any DISC_UNIQUE_BRANCH messages. Test the same branch again for any further responses. If the checksum is invalid then continue branching down the tree, as multiple devices likely exist.
5. When searching the lowest branch (when both sides of the ordered pair are equal) of the tree, send the DISC_MUTE message to that UID. If a device is located at that UID then it will provide a response message and will no longer respond to any DISC_UNIQUE_BRANCH messages.
6. After muting the device, continue searching the unchecked branches by repeating the process.

7.4 Discovery Mute Flag

Each responder shall maintain a Discovery Mute Flag for each responder port. When set, it indicates that a responder will not respond to Discovery Unique Branch messages directed to the UID of that port.

7.4.1 Clearing of Mute Flag

The Mute Flag shall be cleared upon any of the following conditions:

- ☐ Power on, hardware reset, or software reset of responder.
- ☐ Reset command message (RESET_DEVICE) sent to responder.
- ☐ Receipt of DISC_UN_MUTE message addressed to the responder either by using the device's specific UID or through Broadcast Addressing.

7.5 Discovery Unique Branch Message (DISC_UNIQUE_BRANCH)

The following message and response are used for the device discovery process. Discovery messages shall always be addressed to Root Devices (Sub-Device = 0).

A responder shall only respond to this message if its UID is greater than or equal to the Lower Bound UID and less than or equal to the Upper Bound UID included in the message's parameter data, and if it has not been muted through the DISC_MUTE message.

The DISC_UNIQUE_BRANCH message shall always be sent to the ALL_DEVICES_ID UID Address, since all devices must process this message.

Controller:

(Port ID) 0x01 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000
(CC) DISCOVERY_COMMAND	(PID) DISC_UNIQUE_BRANCH	(PDL) 0x0C
(PD)		
Lower Bound UID (48-bit) Upper Bound UID (48-bit)		

Response:

The response message has a number of exceptions to the normal Packet structure to minimize the effect of collisions on legacy devices from appearing as a BREAK, NULL START Code, and data. The format of the DISC_UNIQUE_BRANCH Response Packet is described in Table 7-1.

Controllers shall treat any line activity during the response window as a reply and branch further if necessary, or attempt to Mute a device if an error-free response is received.

The Response to the DISC_UNIQUE_BRANCH message shall not contain a BREAK in the packet. The Responding Device shall reply by enabling the transmitter and simply transmitting the response packet.

No other message structure or normal header information shall be transmitted in the response. To any device monitoring the line, the response will be interpreted as a continuation of the original packet sent from the controller.

All timing information regarding Discovery Responses is included in Section 3.2 Responder Timing Requirements.

Response Packet from Device for DISC_UNIQUE_BRANCH message:

Table 7-1: DISC_UNIQUE_BRANCH Response Packet Encoding

Response Packet Slot	Slot Data		Comments
1	0xFE		Response Preamble bytes that may be dropped by an in-line device during turn-around. Not more than one byte may be dropped by each in-line device.
2	0xFE		
3	0xFE		
4	0xFE		
5	0xFE		
6	0xFE		
7	0xFE		
8	0xAA		Preamble separator byte
9 (EUID11)	Manufacturer ID1 (MSB)	OR with 0xAA	Encoded UID (EUID). Encoding by bit-wise OR with 0xAA and 0x55 as shown.
10 (EUID10)	Manufacturer ID1 (MSB)	OR with 0x55	
11 (EUID9)	Manufacturer ID0 (LSB)	OR with 0xAA	
12 (EUID8)	Manufacturer ID0 (LSB)	OR with 0x55	
13 (EUID7)	Device ID3 (MSB)	OR with 0xAA	
14 (EUID6)	Device ID3 (MSB)	OR with 0x55	
15 (EUID5)	Device ID2	OR with 0xAA	
16 (EUID4)	Device ID2	OR with 0x55	
17 (EUID3)	Device ID1	OR with 0xAA	
18 (EUID2)	Device ID1	OR with 0x55	
19 (EUID1)	Device ID0 (LSB)	OR with 0xAA	
20 (EUID0)	Device ID0 (LSB)	OR with 0x55	
21 (ECS3)	Checksum1 (MSB)	OR with 0xAA	Checksum is the sum of the previous 12 EUID slots. The checksum is an unsigned additive sum of the 8-bit fields into a 16-bit response value.
22 (ECS2)	Checksum1 (MSB)	OR with 0x55	
23 (ECS1)	Checksum0 (LSB)	OR with 0xAA	
24 (ECS0)	Checksum0 (LSB)	OR with 0x55	

The Encoded Unique ID (EUID) is a 12 byte number generated by encoding the UID. It is used solely for devices responding to the Unique Branch Discovery (DISC_UNIQUE_BRANCH) message.

Each byte of the UID shall be encoded into two bytes in the EUID. The purpose of this process is to prevent any combination of colliding EUID data from superimposing in such a way as to generate a properly framed Break – START Code sequence followed by data that some devices might interpret as NULL START Code Data. The response message contains the EUID as outlined in Table 7-1.

Seven extra slots of preamble have been added to the start of the response packet to allow for in-line devices that must shorten the packet for turning around transceivers. If the in-line device shortens the response packet, it shall shorten by exactly one slot time. The controller shall be able to process response packets with 0-7 bytes of preamble.

7.5.1 Response Unique Branch Message Decoding by Controller

To verify integrity of the response, the UID and checksum should be recovered as shown. A valid response shall require the recovered checksum to match that of the EUID.

Table 7-2: DISC_UNIQUE_BRANCH Response Packet Decoding

Decoded Data	Received Encoded Data	Comments
Manufacturer ID (MSB)	EUID11 & EUID10	UID and Checksum are recovered by bit-wise AND (&) as shown.
Manufacturer ID (LSB)	EUID9 & EUID8	
UID3 (MSB)	EUID7 & EUID6	
UID2	EUID5 & EUID4	
UID1	EUID3 & EUID2	
UID0 (LSB)	EUID1 & EUID0	
Checksum (MSB)	ECS3 & ECS2	
Checksum (LSB)	ECS1 & ECS0	

7.5.2 Collisions

Responses to this message may result in collisions. Controllers shall treat detected line activity as a response, indicating the existence of one or more devices on the line within the specified UID range.

7.6 Discovery Mute/Un-Mute Messages

The parameter data area included in the responses to Mute and Un-Mute messages includes a Control Field to inform the controller about specific properties of the device, and may include an optional Binding UID field to indicate the physical arrangement of the responding device.

7.6.1 Control Field

The 16-bit Control Field contains bit flags as in Table 7-3:

Table 7-3: Control Field

Bits 15-4 Reserved (Always set to 0)	Bit 3 Proxied Device Flag	Bit 2 Boot-Loader Flag	Bit 1 Sub-Device Flag	Bit 0 Managed Proxy Flag
--	-------------------------------------	----------------------------------	---------------------------------	------------------------------------

Managed Proxy Flag (Bit 0)

The Managed Proxy Flag (Bit 0) shall be set to 1 when the responder is a Proxy device. See Section 8 for information on Proxy Devices.