

Entertainment Services and Technology Association



Application Guide for ANSI E1.3 - 2001 Entertainment Technology Lighting Control Systems 0 to 10V Analog Control Specification

CP/1999-1008r4

©2001 ASC E1, Safety and Compatibility of Entertainment Technical Equipment and Practices, and its secretariat the Entertainment Services and Technology Association. All rights reserved.

Notice and Disclaimer

ESTA and ANSI Accredited Standards Committee E1 (for which ESTA serves as the secretariat) do 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, or an American National Standard developed under Accredited Standards Committee E1 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 or Accredited Standards Committee E1.

ESTA and ANSI Accredited Standards Committee E1 (ASC E1) neither guaranty nor warrant 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 and ASC E1 do 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 judgement or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstance.

Note: *Draft or proposed standards or recommended practices are subject to change. Conformance to a draft or proposed standard or recommended practice is no assurance that the product or service complies to the final approved standard or practice or any other version thereof.*

Published By:

**Entertainment Services and
Technology Association**

875 Sixth Avenue, Suite 2302
New York, NY 10001 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, PLASA, 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 as Accredited Standards Committee E1, Safety and Compatibility of Entertainment Technical Equipment and Practices.

The Technical Standards Committee (TSC) was established by ESTA's Board of Directors to oversee and coordinate the Technical Standards Program. Made up of individuals experienced in standards-making work from throughout our industry, the Committee approves all projects undertaken and assigns them to the appropriate working group. The Technical Standards Committee employs a Technical Standards Manager to coordinate the work of the Committee and its working groups as well as maintain a "Standards Watch" on behalf of members. Working groups include: Camera Cranes, Control Protocols, Electrical Power, Floors, Fog and Smoke, Photometrics, and Rigging.

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 two 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 manufacturers, consultants, dealers, and end-users. ESTA is committed to developing consensus-based standards and recommended practices in an open setting. Future Control Protocols Working Group projects will include updating this publication as changes in technology and experience warrant, as well as developing new standards and recommended practices for the benefit of the entertainment industry.

Contents

1	Scope	1
2	Signal amplitude	1
3	Diode protection	1
3.1	Limit the slide pot travel	1
3.2	Raise the potentiometer's bottom terminal voltage	1
3.3	Scale the slide pot travel	1
3.4	Include the blocking diode within a feedback loop	2
3.5	Include a compensating diode within a feedback loop	2
4	Cabling	2
5	Connectors and pin assignments	2
6	Troubleshooting hints	5
6.1	Testing controllers	5
6.2	Testing receivers	5
6.3	Testing Cables	5

1 Scope

This document explains some of the design implications that may not be obvious in the E1.3 standard. It also offers some troubleshooting guidance for users of analog control equipment. This document does not add any requirements to the E1.3 standard, nor does it remove any requirements. All the recommendations in this document are advisory only.

2 Signal amplitude

The control signal for this standard is nominally 0 to 10V DC. Section 6.1.1 of E1.3 sets a maximum output range for transmitters of -0.2V to 12.0V. Section 6.2.1 sets the minimum voltage range a receiver must accept without damage as -0.5V to 30.0V. The two different voltage ranges are set so that E1.3 compliant receivers and transmitters will work with each other and also with the vast majority of legacy equipment that is nominally “0 to 10V analog” but that was designed and built before the adoption of E1.3.

The E1.3 transmitter range is set to cover the 0V to 10V scale of meaningful output levels while allowing for manufacturing tolerances and some minor anomalies in output voltage that may result with simple transmitter circuit designs. The 12V maximum is low enough that it is believed not to be likely to damage pre-E1.3 equipment. However, the equipment designer and equipment user should be aware that there is no guarantee that this is the case. Therefore, it is advisable, but not required, to limit the output voltage on E1.3 transmitters to 10.0V.

The E1.3 receiver input voltage acceptance range is set to cover the range of voltages that are likely to be encountered with legacy equipment or if a power supply line is inadvertently connected to a signal line. No guarantee can be made that legacy equipment called “0 to 10V analog” will never impose voltages outside this range.

3 Diode protection

Blocking diodes are required to allow multiple controllers or output devices to be paralleled to control the same dimmers or receiving devices. However, the requirement that blocking diodes be present on the outputs of all E1.3 controllers may cause a dead zone at the low end of control. The dead zone will be caused by forward voltage drop of the blocking diode, which is about 0.6V with silicon diodes. This voltage drop across the diode will cause the output voltage to stay at zero in a dead zone until the fader arm voltage is above 0.6V, a value known as “one diode-drop.” At that point the blocking diode will start to conduct, and the output voltage from the controller will start to rise. This diode-drop must be compensated if smooth low-end fading is desired. For manufacturers of inexpensive controllers, some hints are offered below to compensate for the dead zone:

3.1 Limit the slide pot travel

Since the output will stay at zero until the slide pot reaches a voltage above the diode-drop it is possible to limit the slide pot’s travel such that at the slider’s minimum stop the output has just reached zero.

3.2 Raise the potentiometer’s bottom terminal voltage

Instead of tying the bottom slide pot terminal to common, tie it to some voltage other than common such that the outputs are just at zero when the slide pot is at minimum. A fixed resistor or a diode can be placed between the bottom of the pot and common to raise the bottom of the pot one diode-drop above common.

3.3 Scale the slide pot travel

Print the scale next to the slide pot with “0” where the slide pot just begins to overcome the diode drop, with “5” where the output is five volts (into a specified load), and with “10” where the output is ten volts.

3.4 Include the blocking diode within a feedback loop

For consoles using op-amps, the blocking diode can be placed in the feedback loop of the output driver stage. This has the disadvantage of placing some of the console's circuitry after the blocking diode. A high voltage on the output can therefore damage the console unless additional protection circuitry is added.

3.5 Include a compensating diode within a feedback loop

For consoles using op-amps, a diode of the same type as the blocking diode can be placed in the feedback loop of the output driver stage, while leaving the blocking diode outside the loop. This will introduce an offset into the driver's output that should closely match the forward voltage drop of the blocking diode. It is recommended that a large-value resistor be placed in parallel with the diode to avoid instability.

4 Cabling

Unlike digital and analog multiplex control cables, 0 to 10V cables can be almost any type of conductor or cable. However, the voltage drop in the cable is a major concern. Sections 7.1.1 and 7.1.2 of E1.3 limit the voltage drop in the individual channel conductors and in the common conductor to no more than 0.1V. This limit can be easily exceeded if the current flow in a conductor and the conductor's resistance are not carefully considered.

The common conductor is particularly a concern. If the transmitter is powered from the mains or batteries and supplies all the signal current, the common conductor carries the return current for all the channels. Inattention to the total return current in the conductor and the resistance of that conductor can result in a voltage drop great enough to lower the effective signal voltage and noticeably change the output of the receivers. What is worse, the voltage drop will vary with the signal voltages, which can result in unwanted interaction between channels. For example, a situation can result in which fading up a group of dimmers causes the other dimmers parked at a set level to fade down.

It has often been common practice for an analog lighting controller to be powered from the dimmers it controls, with the power being distributed to the controller in the same cable that carries the individual channel conductors. In this case, the common conductor carries the return current from the controller power supply. Excessive voltage drop in this case will tend to raise the "0V level" at the controller above the "0V level" at the receivers, which can cause "off" control signals to be read at the receivers as something other than off. Dimmers may ghost, or scrollers may fail to go all the way to the first frame.

A simple application of Ohm's Law and some work with a multi-meter can ferret out excessive voltage drop problems in the common conductor and in the individual channel conductors. If necessary, several conductors in parallel or a larger conductor (numerically smaller gauge or larger cross-section) may be used to reduce the voltage drop.

5 Connectors and pin assignments

E1.3 requires labeling to show signal connector pin assignments. It recommends, but does not require, that the pinout should follow a pin number equals channel number configuration with the highest pin number used as signal common.

While no particular connector and no particular pinout is required by E1.3, equipment designers may want to consider using one of the connectors and pin arrangements found on legacy equipment. A selection of these is noted below.

Connector	Typical number of channels	Pinouts	Channels
CHAMP (Centronics)	32	1 - 32 33 & 34 35 & 36	Channels 1-32 Not used Signal common
Cinch Jones 8-pin	6	1 - 6 7 8	Channels 1-6 Power supply (or fan relay) Signal common
Cinch Jones 10-pin	6	1 - 6 7 8 9 10	Channels 1-6 Power supply Signal common Power supply (+15V) Power supply (-15V)
Cinch Jones 10-pin ¹	6	1 - 6 7 8 9 10	Channels 1-6 Not used (or signal common) ² Signal common Not used Not used
Cinch Jones 15-pin	12	1 - 12 13 14 15	Channels 1-12 24V to dimmer for fan relay Power supply Signal common
CPC-28 (Amp)	8	1 - 8 9-24, 27 25 26 28	Channels 1-8 Not used Power supply (+15V) Power supply (-15V) Signal common
DA-15 Strand (DB-15)	12	1 - 12 13 & 14 15	Channels 1-12 Signal common No connection

Note 1: There is an extremely large installed base of 10-pin Cinch-Jones cables made with 8 conductor cable connected to pins 1-8 only.

Note 2: There is a large installed base of 10-pin Cinch-Jones cables made with pins 7 and 8 tied together.

For many years it was the standard practice in the New York metropolitan area to use 10-pin Cinch-Jones plugs to provide six analog control channels. Pins 1-6 were used for channels 1-6. Pins 7 and 8 were used as the common while pins 9 and 10, originally used for a contactor enable, were left unconnected. This led to the use of an 8-conductor cable; however the original installed base of 10-pin plugs on both dimmers and controllers perpetuated this arrangement.

Connector	Typical number of channels	Pinouts	Channels
DB-25 (Artistic Licence, EDI, et alia)	24	1 - 24 25	Channels 1-24 Signal common
DD-50 (DB-50)	48	1 - 48 49 - 50	Channels 1-48 Signal common
DIN 5-pin 180° (N.J.D. Electronics et alia)	4	1 2 3 4 5	Channel 1 Signal common Channel 4 Channel 2 Channel 3
DIN 7-pin (N.J.D. Electronics et alia)	4	1 2 3 4 5 6 7	Channel 1 Signal common Channel 4 Channel 2 Channel 3 +15V power -15Vpower
DIN 8-pin (Zero 88 et alia)	6	1 - 6 7 8	Channels 1-6 Power supply Signal common
DIN 8-pin (Pulsar et alia)	6	1 2 3 - 8	Power supply Signal common Channels 1-6
D-Subminiature 9-pin (Rosco/ET)	6	1 - 6 7 - 9	Channels 1-6 Signal common
Socapex 337P	30	1 - 30 31 - 34 35 - 37	Channels 1-30 Not used Signal common
SRC-16 (Cannon)	12	1 - 11 12 13 14 15 16	Channels 1-11 Power supply (+15V) Power supply (-15V) Not used Signal common Channel 12
XLR 5-pin	4	1 - 4 5 Shell	Channels 1-4 Power supply (+15V) Signal common
XLR 7-pin	6	1 - 6 7 Shell	Channels 1-6 Power supply Signal common

Note: The shell of many XLR-style connectors does not make a good electrical contact. The XLR pinouts listed above are for reference since they have been used in the past. It is not advisable to use these XLR pinouts since they use the shell for the signal common.

6 Troubleshooting hints

6.1 Testing controllers

One advantage of 0 to 10V control over digital or analog multiplex is its simplicity and ease of troubleshooting. Testing a controller or other sending device's output can usually be done with an inexpensive voltmeter.

Place the negative probe on the signal common output pin and the positive probe on the output channel to test. With the output at zero the voltmeter should read zero volts. With the output at full, the voltmeter should read ten volts. With the output at 50%, the voltmeter should read five volts. If the output looks good on the meter but fails to drive the dimmer or other intended controlled device, repeat the measurements with the output loaded with a 20,000 ohm resistor.

6.2 Testing receivers

One simple test of a receiver can be made with a 9V and a 1.5V battery. Connect the negative terminal of the 9V battery to the common input of the receiver. Tie the positive terminal to the channel input you wish to test. The receiver should go to its appropriate response for 90%. If it responds but you can't tell if it's the 90% response, perform the same test with the 1.5 volt battery. The receiver should go to the appropriate 15% response. Unless the receiver has a toggled, two-state response with the trip point above or below these points, the difference between the 90% and 15% responses should be obvious.

A difficult-to-track-down fault may appear to be in the receiver but actually be a problem with the controller/cable/receiver combination. Improperly designed controllers with op-amp outputs may become unstable with reactive loads. The controller output then oscillates at high frequency at some point in the output range, which then can make the receiving device behave bizarrely. A hint that this is the problem is that the symptoms disappear as soon as the control cable is unplugged to start troubleshooting.

6.3 Testing cables

A simple continuity and short circuit test will almost always find a bad analog cable. Unless a cable has been damaged internally by being pulled or by being smashed with a sharp-edged object, intermittent connections are almost always found where the cable enters the connector. Intermittent connections usually can be found by wiggling the connector strain-relief. Maintaining the integrity of the connector strain-relief is important for maintaining reliable analog cables. The electrical connections of the wires should not be relied upon for making the mechanical connection of the analog control cable to the connector.

Contact Information

E1 Secretariat

Entertainment Services and Technology Association (ESTA)

Karl G. Ruling

Technical Standards Manager

875 Sixth Avenue, Suite 2302

New York, NY 10001

Phone: +1-212-244-1505

FAX: +1-212-244-1502

email: kruling@esta.org

Technical Standards Committee Chair

Bill Groener

Fourth Phase

7777 Westside Avenue

North Bergen, NJ 07047

Phone: +1-201-758-4000

FAX: +1-201-758-4312

email: bgroener@fourthphase.com

Control Protocols Working Group Chairs

Steve Carlson

High Speed Design, Inc.

11929 N. W. Old Quarry Road

Portland, OR 97229

Phone: +1-503-626-4206

FAX: +1-503-626-4206

email: scarlson@hspdesign.com

Steve Terry

Fourth Phase

7777 Westside Avenue

North Bergen, NJ 07047

Phone: +1-201-758-4000

FAX: +1-201-758-4312

email: sterry@fourthphase.com

Acknowledgments:

This document was approved by the Control Protocols Working Group on 20 January 2001 at the working group meeting in Irving, Texas. The membership of the working group at the time of the meeting was:

Principal voting members:

John Sellers; AIM Northwest [G]
Tony de Rijk; Amazing Controls! Inc. [P]
Wayne David Howell; Artistic Licence (UK) Ltd. [P]
Tobin Neis; Barbizon Companies [U]
Doug Fleenor; Doug Fleenor Design, Inc. [P]
Ed Jones; Edwin Jones Co., Inc. [P]
Greg Heinzle; Electronic Theatre Controls, Inc. [P]
Tracy Underhill; Electronics Diversified Inc. [P]
Philip Nye; Engineering Arts [G]
Robert Goddard; Goddard Design Co. [P]
Trevor Forrest; Helvar Lighting Control [P]
Scott Blair; High End Systems Inc. [P]
Steve Carlson; High Speed Design, Inc. [P]
Peter Willis; Howard Eaton Lighting Ltd. [P]
Edwin S. Kramer; IATSE, Local 1 [U]
Edward Paget; Jones & Phillips Associates, Inc. [G]
Rick Leinen; NSI / Colortran, representing Leviton Manufacturing Co., Inc. [P]
Ole Pedersen; Martin Professional A/S [P]
Dave Higgins; Pathway Connectivity Inc. [P]
Tim Cox ; Professional Lighting and Sound Association (PLASA) [G]
Steve Terry; Fourth Phase New Jersey, representing Production Resource Group [U]
Robert Barbagallo; Proximo Inc. [U]
Paul F. Mardon; Pulsar Ltd. [P]
Jon R. Farley; Rosco/Entertainment Technology, representing Rosco Laboratories [P]
David George; Spectrum Manufacturing Inc. [P]
Richard Lawrence; Strand Lighting Ltd. [P]
Jerry Gorrell; Theatre Safety Programs [G]
Brian Dowd; TMB Associates [U]
Mitch Hefter; Rosco/Entertainment Technology, representing USITT [U]
Charles Reese; Vari-Lite, Inc. [P]
Eckart Steffens; SOUNDLIGHT, representing The Professional Lighting and Sound Association of Germany (VPLT) [G]
Richard Thornton Brown; Zero 88 Ltd. [P]

Alternate voting members:

Milton Davis; Doug Fleenor Design, Inc. [P]
Ed Prasser; Electronic Theatre Controls, Inc. [P]
Scott Rempel; Electronics Diversified Inc. [P]
Dexter McNeil; Goddard Design Co. [P]
John Huntington; IATSE Local 1 [U]
Ken Vannice; NSI / Colortran, representing Leviton Manufacturing Co., Inc. [P]
Michael (Sandy) Twose; Pathway Connectivity Inc. [P]
Tony Douglas-Beveridge; Professional Lighting and Sound Association (PLASA) [G]
George Sabbi; PRG Lighting and Audio Group [U]
George Kindler; Thoughtful Designs, representing Production Resource Group [U]
Steve Unwin; Pulsar Ltd. [P]
Peter Brooks; Zero 88 Ltd. [P]

[G] = general interest

[P] = producer

[U] = user

Observer members:

Steve Friedlander; Auerbach & Associates, Inc. [U]
Sierk Janszen; Avenger Showcontrol [P]
J. B. Toby; Avolites Ltd. [P]
Shahid Anwar; Avolites Ltd. [P]
Richard Salzedo; Avolites Ltd. [P]
Barbara Wohlsen; [U]
Bernardo Benito Rico; Ben-Ri Electronica S.A. [P]
Lee J. Bloch; Bloch Design Group Inc. [G]
Soo-Myong Chung; Bloch Design Group Inc. [G]
Bradley Klinkradt; [G]
Ted Fregon; Bytecraft Pty. Ltd. [P]
Murray Mason; Bytecraft Pty. Ltd. [P]
Bill Ellis; Candela Controls, Inc. [U]
Marty Lazarus; Chicago Spotlight, Inc. [G]
Tal Miron; Compulite R & D [P]
Jason Friedman; Creative Realities, Inc. [G]
Mikael Fahl; Dataton AB [P]
David Bertenshaw; [P]
Gary Dove; Dove Systems [P]
Jussi Kallioinen; Eastway Sound & Lighting [U]
Bill Fehrmann; Electrol Engineering, Inc. [P]
Adam Bennette; Electronic Theatre Controls Ltd. [P]
Paul Bennett; Electronics Diversified Inc. [P]
Paul K. Ericson; Ericson Lighting Design [U]
Gerald Jones; [U]
Roger Lattin; IATSE Local 728 [U]
Rob Johnston; Interactive Technologies, Inc. [P]
Larry Schoeneman; Interesting Products, Inc. [G]
David Timmins; Jands Electronics [P]
Merlin Milner; Leax Lighting Controls [P]
John Mehltrittter; Lehigh Electric Products Co. [P]
Mark T. Kraft; Lehigh Electric Products Co. [P]
Andrew Sherar; Lightmoves PLC [P]
Gary Pritchard; LSC Lighting Systems PTY Ltd. [P]
J. P. Steiner; Lutron Electronics [P]
Tracy Schwenk; Martin Professional Inc. [P]
Gerard Cohen; Martin Professional Inc. [P]
Hiroshi Kita; Marumo Electric Co., Ltd. [P]
Geoffrey O. Thompson; IEEE 802.3 / Nortel Networks [G]
David A. Boller; Organic Machines LLC [P]
William Benner; Pangolin Laser Systems [P]
Mac Perkins; PNTA Inc. [G]
Stephen J. Tyrrell; Quantum Logic [P]
Charlie Richmond; Richmond Sound Design Ltd. [P]
Kenneth Mockler; Sceno Plus [U]
Mick Martin; Showcad Control Systems [P]
Andre Broucke; ADB - TTV Technologies, representing Siemens [P]
Ujall Kar; Standard Robotics & Lighting [G]
John Mitchell; Sterling Technical Services [U]
Tad Trylski; [U]

Thom Weaver; [G]
Colin Waters; TMB Associates [U]
Hans Leiter; Transtechnik GmbH [P]
Anders Ekvall; Transtechnik GmbH [P]
Ken Wagner; Walt Disney Imagineering [U]
Eric Cornwell; West Side Systems [U]
Keny Whitright; Wybron, Inc. [P]
John Sondericker III; Wybron, Inc. [P]

[G] = general interest

[P] = producer

[U] = user

Entertainment Services and Technology Association

United States Institute for Theatre Technology, Inc.

DRAFT
BSR E1.11, Entertainment Technology -
USITT DMX512-A
Asynchronous Serial Digital Data Transmission Standard
for
Controlling Lighting Equipment and Accessories

Revision 3

© 1999, 2000 Entertainment Services and Technology Association
© 1999, 2000 United States Institute for Theatre Technology, Inc.

**This document is a work in progress, and may be duplicated only for the purposes of finalizing this Report.
It may not be published in part or in whole or be duplicated for-profit or sold in any manner.**

Reminder -- as noted above, ESTA 'Work in Progress' documents are copyrighted and only may be copied for the purpose of developing the Standard within the Task Group or Working Group the project is assigned to. It is the policy of the ESTA Technical Standards Program that publishing of such preliminary information, such as in this document, in any form, including on Web Pages, is not allowed.

NOTICE and DISCLAIMER

ESTA and ANSI Accredited Standards Committee E1 (for which ESTA serves as the secretariat) do 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, or an American National Standard developed under Accredited Standards Committee E1 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 or Accredited Standards Committee E1.

ESTA and ANSI Accredited Standards Committee E1 (ASC E1) neither guaranty nor warrant 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 and ASC E1 do 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 judgement or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstance.

Note: *Draft or proposed standards or recommended practices are subject to change. Conformance to a draft or proposed standard or recommended practice is no assurance that the product or service complies to the final approved standard or practice or any other version thereof.*

DRAFT Standard, BSR E1.11, Entertainment Technology – USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories

Foreword	vii
Introduction	viii
1 General	1
1.1 Scope	1
1.2 Overview and Architecture	1
1.3 Appropriate uses of this Standard	2
1.4 Classes of data appropriate for transmission over links designed to this Standard	2
1.5 Classes of data not allowed on this Standard	2
1.6 Compliance	2
2 Normative references	3
3 Definitions	4
4 Electrical Specifications and Physical Layer	6
4.1 General	6
4.2 Electrical isolation	6
4.3 Topology	6
4.4 Ports	6
4.5 Data link common and grounding topologies	6
4.6 Preferred method of earth grounding data link common	7
4.7 Primary data link	7
4.8 Optional secondary data link (Enhanced Functionality)	7
4.9 Data Link signal designations summary	7
4.10 Data termination procedures	8
5 Nominal Operating Characteristics	9
5.1 General	9
5.2 Earth grounding of data link common for transmitters	9
5.3 Transmitter characteristics	9
5.4 Ground referenced transmitters	10
5.5 Earth grounding of data link common for receivers	11
5.6 Isolated Receiver characteristics	11
5.7 Disallowed receiver topologies	13
5.8 Processing devices	13
5.9 Loading designation	13
6 Protection	14
6.1 Minimum protection against interconnection damage	14
6.2 Minimum Electro Static Discharge (ESD) protection	14
7 Connection Methods	15
7.1 Equipment fitted with external user accessible pluggable data link connections	15
7.2 Equipment intended for fixed installation with internal connections to the data link	15
7.3 Passive Data Outlets or Wall plate panels	15
7.4 RJ45 type connectors	16

DRAFT Standard, BSR E1.11, Entertainment Technology – USITT DMX512-A Asynchronous Serial Digital
Data Transmission Standard for Controlling Lighting Equipment and Accessories

8 Cable	17
8.1 Background	17
8.2 General	17
8.3 General Applications and between all Portable Equipment	17
8.4 Cable between permanently installed fixed equipment	17
8.5 Cable application rules	18
9 Data Protocol	19
9.1 Format	19
9.2 Slot format	19
9.3 Break	19
9.4 Mark After Break	19
9.5 START Code	19
9.6 Maximum number of data slots	20
9.7 Minimum number of data slots	20
9.8 Defined line state between slots	20
9.9 Defined line state between data packets (Mark Before Break)	20
9.10 Break-to-Break spacing	20
9.11 Dimmer class data	21
9.12 Timing Diagram - output of transmitting UART	21
10 Receiver Performance	23
10.1 Loss of data tolerance / Resumption of acceptance of data	23
10.2 Receiver performance at maximum update rate	23
10.3 Inactive receiver input circuitry	23
10.4 Packet processing latency	23
11 Marking and Disclosures	24
11.1 Identification	24
11.2 Port marking	24
11.3 Data line termination marking	24
11.4 Ground / Isolation marking	25
11.5 Required disclosures	25
Annex A - Alternate (not preferred) topologies (Normative)	27
A1 Isolated transmitters	27
A2 Non-isolated receivers	28
A3 Grounded Receivers	29
A4 Earth grounding of data link common for floating devices	30

DRAFT Standard, BSR E1.11, Entertainment Technology – USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories

Annex B (Normative) - Enhanced DMX512, the second data link	33
B1 General	33
B2 Summary of Enhanced Function Topologies	33
B3 Identification of data protocols on the secondary data link	34
B3.1 EF1- Full duplex DMX512	34
B3.2 EF2 Half duplex on the second data pair	34
B3.3 EF3 Protocols that use the primary and the secondary data links in ways not covered above	35
Annex C (Informative) - Higher Protection Levels	37
Annex D (Normative) - Alternate START Codes Implementation	39
D1 ASC refresh interval	39
D2 Timing differences for Alternate START Code packets	39
D3 Handling of Alternate START Code packets by in-line devices	39
Annex E (Normative) - Reserved Alternate START Codes	41
E1 Reserved Alternate START Codes	41
E2 ASC text packet	41
E3 ASC test packet	42
E4 System Information Packet (SIP) Alternate START Code	42
E4.1 Application	42
E4.2 SIP format	43
E4.3 Control bit field	44
E4.4 Checksums	44
E4.5 SIP Sequence number	44
E4.6 Originating universe	44
E4.7 DMX512 processing level	44
E4.8 Software version	44
E4.9 Packet lengths	44
E4.10 Number of packets	45
E4.11 Manufacturer ID	45
E4.12 Packet history	45
E4.13 SIP Checksum	45

DRAFT Standard, BSR E1.11, Entertainment Technology – USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories

Annex F (Normative) - Alternate START Code, Manufacturer ID, and Enhanced Functionality Registration	47
F1 Alternate START Code Registration Policy: 1 - 255 decimal (01 - FF hexadecimal).	47
F2 Authorized use	47
F3 Reserved Alternate START Codes	47
F4 Requests for Registration of New START Codes	47
F.4.1 Number of Alternate START Codes per entity	47
F.4.2 Selection of the Alternate START Code value and Manufacturer ID	47
F5 Requirements for registration of an EF protocol	47
F6 Documentation Register	48
F6.1 Documentation for use of Alternate START Codes	48
F6.2 Maintenance and Publication	48
F6.3 Supplemental documentation	48
F7 Ownership	48

Tables:

1 - Signal designations summary	7
2 - Ground Referenced Transmitter characteristics	10
3 - Isolated Receiver characteristics	11
4 - Connection Schedule for DMX512 systems using ISO IEC 11801 Cable	16
5 - Cable application rules	18
6 - Data slot format	19
7 - Timing Diagram Values	22
8 - Signal designations abbreviations allowed for marking	24
9 - Ground / Isolation marking	25
10 - 5-pin XLR data outlet / wall plate marking	26
A1 - Isolated Transmitter characteristics	28
A2 - Non-Isolated Receiver characteristics	29
B1 - Enhanced Function topologies	33
C1 - Transmitter protection	37
C2 - Receiver protection	37
E1 - Reserved Alternate START Codes	41
E2 - SIP Format	43

Figures:

1 - Ground referenced transmitter	10
2 - Isolated receiver	12
3 - Receiver topology not allowed	13
4 - Timing diagram	21
A1 - Isolated transmitter	27
A2 - Non-isolated receiver	28
A3 - Grounded receiver	30
A4 - DMX512 device, Floating	31

Foreword

The **Entertainment Services and Technology Association** (ESTA) is a non-profit trade association representing the North American entertainment technology industry. Its members include dealers, manufacturers, manufacturer representatives, service and production companies, scenic houses, designers and consultants. The Association addresses areas of common concern such as technical standards, customer service, equipment quality, business practices, insurance, and credit reporting, and provides a wide variety of services to Members.

ESTA's Technical Standards Committee (TSC) is accredited by the American National Standards Institute (ANSI) as Accredited Standards Committee *E1, Safety and Compatibility of Entertainment Technical Equipment and Practices* with ESTA as its Secretariat. This accreditation means that the ESTA Technical Standards Program for standards-making has passed a detailed scrutiny by ANSI to insure that it meets the most stringent requirements for fairness and proper public review of proposed ESTA standards. The accreditation allows ESTA to submit standards for the ANSI public review and comment process, and then publish them as ANSI standards. The ESTA Technical Standards Program is now the only ANSI-accredited standards-making program dedicated to the needs of entertainment technology.

ESTA
875 6th Ave - Suite 2302
New York, NY 10001

(212) 244-1505(212) 244-1502 FAX

<http://www.esta.org>

The **United States Institute for Theatre Technology, Inc.** (USITT) is the Association of Design, Production, and Technology Professionals in the Performing Arts and Entertainment Industry. Founded in 1960, the mission of the Institute is to advance the professions of design and technology in the performing arts by disseminating information, actively promoting the advancement of knowledge and skills and facilitating national and international communication among its members. USITT is the United States Center of OISTAT, the International Organization of Scenographers, Theatre Architects and Technicians.

USITT
6443 Ridings Rd.
Syracuse, NY 13206-1111

(800) 93USITT(315) 463-6463(315) 463-6525 FAX

<http://www.usitt.org>

© 1999, 2000 Entertainment Services and Technology Association
© 1999, 2000 United States Institute for Theatre Technology, Inc.

This document is a work in progress, and may be duplicated only for the purposes of finalizing this Report. It may not be published in part or in whole or be duplicated for-profit or sold in any manner.

1 Introduction

2
3 The original version of the DMX512 Standard was developed in 1986 by the Engineering Commission of the
4 United States Institute for Theatre Technology, Inc. (USITT). Minor revisions were made in 1990. DMX512 has
5 gained international acceptance throughout the entertainment industry, even though USITT is not formally
6 accredited as a standards making body. The earlier versions of this Standard covered only data used by
7 dimmers. In practice this Standard has been used by a wide variety of devices; this version recognizes this fact.
8

9 In 1998, it became evident that additional updates to the Standard were necessary and formal recognition through
10 an internationally recognized standards organization was required. The USITT DMX512 Subcommittee issued
11 a Call for Comments in order to solicit recommendations for changes to the Standard. At the same time, USITT
12 transferred maintenance of DMX512 to the Entertainment Services and Technology Association (ESTA), which
13 is the secretariat for the ANSI Accredited Standards Committee *E1, Safety and Compatibility of Entertainment*
14 *Technical Equipment and Practices* (more commonly known as the ESTA Technical Standards Program - TSP).
15

16 A Task Group established under the ESTA TSP's Control Protocols Working Group acted on the proposals
17 received in response to the Call for Comments. The primary goal was to make editorial updates to DMX512
18 appropriate for current times, including the addition of technical features while maintaining a balance with
19 backward compatibility. Many proposals, while technically innovative, could not be accepted because their
20 implementation would not have been backward compatible and would have immediately rendered obsolete most
21 of the installed base of equipment.
22

23 This document is a result of the actions taken on those proposals and subsequent development under the
24 *Policies and Procedures* of the ESTA Technical Standards Program.
25
26

1 General

1.1 Scope

This Standard describes a method of digital data transmission between controllers and lighting equipment and accessories, including dimmers. It covers electrical characteristics, data format, data protocol, connector type, and recommended cable types.

This Standard is intended as a guide for:

1. Equipment manufacturers and system specifiers who wish to integrate systems of lighting equipment and accessories, including dimmers, with controllers made by different manufacturers.
2. Equipment manufacturers seeking to implement a standard digital transmission protocol in their lighting control and accessory products.
3. System specifiers and consultants to gain detailed information about recommended cable types and allowed connectors.

This standard is not intended to replace existing protocols other than USITT DMX512 and DMX512/1990.

Equipment compliant with this standard will be marked DMX512-A in order to distinguish it from the previous (informally recognized) versions. Unless otherwise noted, references to DMX512 in this document refer to DMX512-A.

1.2 Overview and Architecture

This standard uses a simple asynchronous eight-bit serial protocol consisting of an untyped byte stream produced by standard UARTs. The physical media is normally, but not exclusively, a two-pair cable, with each pair serving as a data link. The media is driven using ANSI/TIA/EIA-485-A-1998 (hereafter referred to as EIA-485-A in this document) balanced data transmission techniques. Physical connection at devices is via 5-pin XLR connectors or by “hard-wiring” to terminals.

The first pair of wires in any DMX512 data cable is used as the primary data link. The second pair, when implemented, is used for a variety of purposes, all of which fall within the scope of EIA-485-A, but not defined in this standard. Identification of the required circuit topology for any particular implementation is defined.

Certain exceptions to the above cabling and connection schemes are permitted where detailed in this standard.

Data on the primary data link is sent in packets of up to 513 slots. The first slot is a START CODE, which defines the information in the subsequent slots in the packet. The interoperability of equipment complying with the standard is largely due to the use of the NULL START Code by transmitting devices. A NULL START Code is one that contains eight data bits all of zero value (a NULL byte) and indicates that all subsequent data within the packet is sequentially numbered bytes of uncategorized data. Proper function is dependent upon the receiving device(s) isolating the pertinent data for processing from each transmitted packet.

1 **1.3 Appropriate uses of this Standard**

2 Equipment designers and general users of this Standard must recognize that this Standard is intended to fill only
3 a limited range of uses. Other standards will be more appropriate for different uses. This is not intended to
4 support a venue wide network that can carry data for lighting, sound, and scenery mechanization, for example,
5 all on the same wire.

6
7 This Standard performs no error checking of NULL START Code packets. There is no assurance that all
8 DMX512 packets will be delivered. It is common practice for merge units and protocol convertors to drop packets
9 that they cannot process in a timely manner. The 1986 and 1990 versions of the USITT Standard specifically
10 allow dimmers to ignore packets that they cannot process in a timely manner, and this concept survives in this
11 version of the Standard with respect to NULL START Code packets.

12
13 **1.4 Classes of data appropriate for transmission over links designed to this Standard**

14 DMX512 is designed to carry repetitive control data from a single controller to one or more receivers. This
15 protocol is intended to be used to control dimmers, other lighting control devices and related non-hazardous
16 effects equipment.

17
18 **1.5 Classes of data not allowed on this Standard**

19 DMX512 is not an appropriate control protocol for hazardous applications, including but not limited to Pyrotechnic
20 Control and Scenery Mechanization.

21
22 Data that controls any device that has a reasonable potential to cause serious physical injury shall not be
23 transmitted over data links built to this Standard. No one shall make, market, or sell such a system while claiming
24 to be DMX512 or DMX512 compatible or any similar such wording.

25
26 **1.6 Compliance**

27 Compliance with this Standard is strictly voluntary and the responsibility of the manufacturer. Markings and
28 identification or other claims of compliance do not constitute certification or approval by the E1 Accredited
29 Standards Committee. See clause 11 for Marking and Disclosure requirements.

2 Normative references

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.

issued by:

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

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

and available from: Global Engineering Documents
15 Inverness Way East
Englewood, CO 80112 USA
Phone: +1-800-854-7179 Fax: +1-303-397-2740 <http://global.ihs.com/>

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 11801 *Information Technology – Generic cabling for customer premises*

ISO/IEC 646 *Information Technology - ISO 7-bit Coded Character Set for Information Interchange*

USITT DMX512/1990 *Digital Data Transmission Standard for Dimmers and Controllers*

issued by and available from: USITT
6443 Ridings Rd.
Syracuse, NY 13206-1111
+1-800-938-7488 +1-315-463-6463 Fax: +1-315-463-6525
<http://www.usitt.org>

3 Definitions

- 3.1 Asynchronous:** signals that start at any time and are not locked or synchronized to the receiving device by a separate clock line.
- 3.2 Balanced Line:** a data communications line where two wires are present, the signal and its opposite (complement), the actual signal being the difference between the voltages on the two wires. Balanced lines have excellent noise and interference rejection properties.
- 3.3 Break:** a high to low transition (space) followed by a low of at least 88 microseconds followed by a low to high transition.
- 3.4 Common:** see Data Link and Signal Common.
- 3.5 Common-Mode Voltage:** a voltage appearing equally on the data + (plus) and data - (minus) lines relative to signal common. $V_{cm} = (V_a + V_b)/2$ where:
 V_{cm} is the Common Mode Voltage
 V_a is the voltage on DMX512 data + with respect to signal common
 V_b is the voltage on DMX512 data - with respect to signal common
- 3.6 Controller:** a transmitting device that originates DMX512 data.
- 3.7 Data +:** true signal.
- 3.8 Data -:** complementary data signal.
- 3.9 Data Link:** physical connection between transmitting and receiving devices.
- 3.10 Data Link Common:** the connection to signal common at the point of interconnection (DMX512 Port) of the product.
- 3.11 DMX512 Port:** see Port.
- 3.12 DMX512 Processing Device:** a piece of equipment that regenerates the timing of any DMX512 packet or has provision for other signal inputs from which the outgoing DMX512 packet is generated. In the absence of any DMX512 transmitting capability, the device has provision for other signal outputs which are controlled in some manner by the incoming DMX512 packet. Basic buffer products are not normally considered processing devices.
- 3.13 Driver:** the circuit which drives the transmit signal and is directly connected to the DMX512 line. See Line Driver.
- 3.14 Enhanced Functionality:** Active use of the optional secondary data link of a DMX512 port.
- 3.15 Idle:** the time that the DMX512 line is high and not sending any information (also known as the 'Mark' condition).
- 3.16 In-Line Device:** any component that receives and re-transmits DMX512.
- 3.17 Isolation:** circuit topology in which the output is completely electrically disconnected from the input.
- 3.18 Isolation voltage:** voltage specification between input and output stages of an isolated system at or below which damage or breakdown of circuit components will not occur.
- 3.19 Legacy Equipment:** transmitting and receiving devices complying with the original USITT DMX512 or DMX512/1990 in all aspects of those standards. (Exception: receiving devices that are not dimmers but comply with all other aspects of DMX512/1990 shall be considered to be Legacy Equipment.)
- 3.20 LEN:** (Load Equivalent Number) the number or fractions of Unit Loads as defined by EIA-485-A.
- 3.21 Line Driver:** an electrical circuit providing differential voltage excursions on a data link, operating within a defined Common Mode voltage range and with a specified response to overload and overvoltage conditions.
- 3.22 Line Receiver:** an electrical circuit allowing detection of differential voltage excursions on a data link, operating within a defined Common Mode voltage range and with a specified response to overload and overvoltage conditions.
- 3.23 Loop-Through Connection:** a connector or terminal port which connects the signals present on at least Pins 1, 2 and 3 of one port to another port. Frequently abbreviated to Loop or Thru.

- 1 **3.24 Manufacturer ID:** a two-byte value assigned to a Manufacturer/Organization by the E1 Accredited
2 Standards Committee for use with Alternate START Codes 91h and CFh. This ID serves as an identifier
3 that the data following in that packet is proprietary to that entity and should be ignored by all others.
- 4 **3.25 Mark:** a line condition where Signal True is high with respect to Signal Complement.
- 5 **3.26 MAB (MaB):** Mark After Break – the period of time between the low to high transition which signifies the
6 end of Break and the high to low transition which is the start bit of the START Code (slot 0).
- 7 **3.27 MBB (MbB):** Mark Before Break – the period of time between the end of the last stop bit of the last slot and
8 the high to low transition which signifies the start of Break.
- 9 **3.28 Merge Unit:** a product comprising one or more receiving devices and one or more transmitting devices that
10 generate a DMX512 packet derived from the manufacturers declared logical combination of the DMX512
11 input packets.
- 12 **3.29 NULL START Code:** a START Code with a value of 00h.
- 13 **3.30 Packet (in DMX512-A):** a Reset Sequence followed by all slots up to a Mark Before Break and the start of
14 the next Reset Sequence.
- 15 **3.31 Port:** a DMX512 signal connection point (connector or terminal strip).
- 16 **3.32 Receiver:** see Line Receiver.
- 17 **3.33 Receiving Device:** a piece of equipment which accepts a DMX512 signal.
- 18 **3.34 Reset:** see Break.
- 19 **3.35 Reset Sequence:** a sequence of a Break, Mark After Break, and START Code.
- 20 **3.36 Signal Common:** the common reference (zero volt supply) of the EIA-485-A driver or receiver circuitry.
- 21 **3.37 Slot:** a sequentially numbered framed byte in a DMX512 packet. A single Universe contains a maximum
22 of 513 Slots, starting at slot 0. Slot 0 is the START Code.
- 23 **3.38 Slot Footprint:** the number of slots of data used by a product in its operation.
24 *Note: A 24 way dimmer rack may have a footprint of 24, it may be more if some slots are used to provide*
25 *additional control functions using NULL START Code packets. Automated fixtures usually require a Slot*
26 *Footprint of greater than one.*
- 27 **3.39 Space:** a line condition where Signal True is low with respect to Signal Complement.
- 28 **3.40 Start Bit:** the extra bit attached to the beginning of a byte to indicate to the receiver that a new byte is being
29 sent. The start bit is always low, i.e., Space.
- 30 **3.41 START Code:** the first slot sent after Break, indicating the type of information to follow.
- 31 **3.42 Stop Bit:** the extra bit(s) attached to a byte to indicate the end of the byte – DMX512 has 2 stop bits. The
32 stop bit is always high, i.e., Mark.
- 33 **3.43 Terminator:** a device or circuit topology that is designed to minimize unwanted signal reflections on a data
34 link.
- 35 **3.44 Transmitting Device:** a piece of equipment that produces a DMX512 signal.
- 36 **3.45 UART:** Universal Asynchronous Receiver/Transmitter
- 37 **3.46 Universe:** a DMX512 data link originating from a single DMX512 source. Control of up to 512 DMX512
38 data slots comprises a single universe.
- 39 **3.47 Update (Refresh) Rate:** the number of DMX512 packets with a NULL START Code sent per second.
- 40

4 Electrical Specifications and Physical Layer

4.1 General

Except where specifically called out in this document the electrical specifications of this Standard are those of EIA-485-A. Where a conflict between EIA-485-A and this document exists, this document is controlling as far as this Standard is concerned.

The physical layer of a DMX512 data link is constrained by earth grounding practices, termination methods, signal levels, EMC, and accidental damage by connection to other devices.

Equipment designers shall pay particular attention to EIA-485-A requirements for line drivers, line receiver design, line voltage levels, line loading, and the Common Mode requirements of EIA-485-A. Further, equipment designers must comply with the non EIA-485-A requirements of clauses 5 and 6.

4.2 Electrical isolation

EIA-485-A makes no general provisions for electrical isolation. However, this Standard does, and suitable optical isolation, transformer isolation, or other means may be employed to prevent the undesirable propagation of voltages which exceed the Common Mode limits of EIA-485-A (see clauses 5 and 6). The inclusion of such isolation does not, however, alter the requirement that a transmitter or receiver shall conform to EIA-485-A.

4.3 Topology

A data link shall consist of a single active differential line driver, a terminated transmission line and one or more differential line receivers. The transmission line shall be a data cable with a nominal characteristic impedance of 100 to 120 ohms as specified in clause 8. The differential drivers and receivers shall meet the requirements of EIA-485-A and all additional requirements of this Standard.

4.4 Ports

A DMX512 Port is the DMX512 signal connection point between the internal electronics of a device and the physical media of the transmission line. It may be made either by the prescribed connector as defined in clause 7 or by a terminal strip. Terminals and connector contacts are referred to as Pins in this Standard. A DMX512 port has five pins, designated 1 through 5.

Pin 1 of a DMX512 Port is the signal common. All DMX512 ports shall connect Pin 1 to the signal common on the physical media.

4.5 Data link common and grounding topologies

Various paragraphs of clause 5 and Annex A deal with shield-to-earth grounding topologies. In all cases there shall be a low impedance connection between Pin 1 of the DMX512 port and signal common of the EIA-485-A driver or receiver circuitry.

4.6 Preferred method of earth grounding data link common

DMX512 systems should make use of earth ground referenced transmitting devices and isolated receiving devices. This approach provides for a single point solid ground/chassis connection at the source, and allows for variations in building ground potentials between transmitting and receiving devices. This is to ensure that interoperability of equipment is achieved in situations that do not in their own right constitute a safety hazard, but might otherwise exceed the Common Mode limitations of EIA-485-A. See EIA-485-A clause 4.3.1. Other approaches are covered in Annex A.

4.7 Primary data link

Pins 2 and 3 of a DMX512 Port form the primary data link. Pin 2 is the data - signal. Pin 3 is the data + signal. Format of the data is covered in clause 9. Limited use of multiple data link drivers for half-duplex, bi-directional data transmission on the primary data link is permitted in accordance with Enhanced Functionality as described in Annex B.

4.8 Optional secondary data link (Enhanced Functionality)

4.8.1 Secondary data link - active use

Pins 4 and 5 of a DMX512 Port provide a secondary EIA-485-A data link. Implementation of this data link is optional. Active use of Pins 4 and 5 is known as Enhanced Functionality. Pin 4 is the data - signal. Pin 5 is the data + signal. Several different network topologies are associated with the implementation of Enhanced Functionality. Approved uses of the secondary data link and their associated topologies are described in Annex B.

4.8.2 Secondary data link - passive use

In order to extend Enhanced Functionality across a network, devices containing two ports for receive and transmit that do not actively process data on Pins 4 and 5 must provide a direct passive link of these pins between the two ports. Devices containing three or more ports may wire the passive secondary data link between only two of the ports. These two ports shall be declared as required in Clause 11 and should be marked on the product.

4.9 Data Link signal designations summary

Table 1 - Signal designations summary

Function	Pin Reference within Standard	DMX512 Function
Data Link Common	1	Common (Screen)
Primary Data Link	2	Data 1 -
	3	Data 1 +
Secondary Data Link (Optional - see clause 4.8)	4	Data 2 -
	5	Data 2 +

While the generic reference to Pins 1 through 5 correlates to the physical pinout used on the XLR style connectors as defined in clause 7 of this Standard, there are other situations where different physical pinouts may be encountered, such as a terminal strip in a fixed installation with internal connections.

4.10 Data termination procedures

DMX512 data links shall be terminated to eliminate ringing and signal reflection which can cause mis-operation of an otherwise properly designed system. To comply with this Standard, all equipment connected to a DMX512 data link shall operate in accordance with the stated manufacturer's specification when the data link is terminated. The impedance of this terminator shall be equal to the characteristic impedance of the transmission line – 120 ohms, optionally in series with a 0.047µf capacitor between Data + and Data -.

Receiving devices without a loop-through connection shall be permanently terminated. Receiving devices with loop-through shall not be permanently terminated.

In the preferred topology where the transmitter is connected to one end of the data link, the far end of the data link shall be terminated. In the case where the transmitter cannot be connected at one end of the data link, then both ends of the data link shall be terminated.

Manufacturers of receiving devices may provide internal termination of the data link. Where such termination is provided, it shall comply with the electrical and marking requirements of this Standard. It is recommended that termination components be chosen to withstand continuous voltages of at least 30 VAC/42 VDC.

Unpowered connected DMX512 devices shall not degrade the performance of the DMX512 transmission system.

5 Nominal Operating Characteristics

5.1 General

Operation limits generally follow the detailed requirements of EIA-485-A for generator characteristics. Where appropriate, separate limits are given for isolated products.

5.2 Earth grounding of data link common for transmitters

In recognition of the need for DMX512 compliant product to be capable of interconnection as part of large and potentially complex systems, this Standard defines two allowable topologies for the earth grounding of data link common and signal common for transmitters, to be known as “Ground Referenced” and “Isolated”. The preferred method is “Ground Referenced.” “Isolated” is covered in Annex A.

5.3 Transmitter characteristics

All electrical characteristics shall be measured at the output terminals of the product. Transmitting devices shall deliver open circuit output voltage not less than 1.5V and not more than 6V as defined in EIA-485-A clause 4.2.1. Transmitting devices shall comply with the requirements of EIA-485-A clause 4.2.2 for differential and offset output voltages, which requires that the magnitude of the differential output voltage be not less than 1.5V and not more than 5V when connected to a test load of 54 ohms.

The requirements of EIA-485-A clause 4.2.2 for generator offset voltage and EIA-485-A clause 4.2.3 for Differential Output voltage (Common Mode loading) shall be met. The requirements of EIA-485-A clause 4.2.4 for Off-state output current shall be met, provided that the Unit load for the generator in the off state does not exceed 1.

Transmitting devices shall comply with the requirements of EIA-485-A clause 4.2.7, for which the unit interval (t_{ui}) shall be regarded as 4 microseconds.

In battery operated equipment or equipment which has no inherent provision for connection to protective ground, chassis shall be deemed to be any exposed metal connector parts which do not carry signals. All such parts shall be at equal potential.

5.4 Ground referenced transmitters

Ground referenced transmitting device outputs shall meet the following conditions in table 2 during normal operation under open circuit condition.

Table 2 - Ground Referenced Transmitter Characteristics

Connection	Limit	Comment
Pin 2 to Pin 1 or Pin 3 to Pin 1	$0 \leq v \leq +6 \text{ VDC}$	
Pin 4 to Pin 1 or Pin 5 to Pin 1	$0 \leq v \leq +6 \text{ VDC}$	Enhanced Function devices only
Pin 1 to Chassis	0V	Ground Referenced transmitting devices shall have a direct connection between Pin 1 and chassis.
Pin 2 to Chassis or Pin 3 to Chassis	$0 \leq v \leq +6 \text{ VDC}$	
Pin 4 to Chassis or Pin 5 to Chassis	$0 \leq v \leq +6 \text{ VDC}$	Enhanced Function devices only
Pin 2 to Pin 3	+/- 6V (open circuit)	
Pin 4 to Pin 5	+/- 6V (open circuit)	

Figure 1 illustrates a ground referenced transmitter port. It is characterized by the direct connection of the shield (Pin 1) to chassis and protective earth. Therefore, devices employing ground referenced transmitters shall be provided with provision for connection to protective earth. Any impedance (A) between Pin 1 and zero volt supply of the transmitter circuit shall be less than 100 ohms. Any impedance (B) between Pin 1 and chassis shall be less than 20 ohms and is preferably zero ohms.

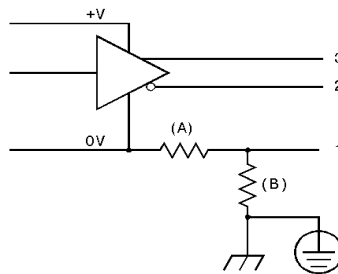


Figure Key

- 1 - DMX512 Pin 1
- 2 - DMX512 Pin 2 (or Pin 4)
- 3 - DMX512 Pin 3 (or Pin 5)
- A - Optional Impedance (see text)
- B - Optional Impedance (see text)

Figure 1 - Ground Referenced Transmitter

A DMX512 device may have any number of Grounded transmitter ports. Grounded transmitter ports may be used by all DMX512 devices including ones that provide any number of non-DMX512 input or output ports. Adherence to this topology allows a DMX512 transmitter connector to be marked as GROUNDED.

Because the transmitter in this topology is grounded, the existence of an isolation barrier between the transmitter and any other part of the device shall NOT qualify output for marking as ISOLATED.

5.5 Earth grounding of data link common for receivers

This Standard defines several allowable topologies for earth grounding of data link common and signal common for receiving devices. These are to be known as “non-isolated” and “isolated”. The preferred method is “isolated.” A specific concession is available to manufacturers of non-isolated receivers who, for reasons beyond the scope of this Standard, require a direct link between data link common and chassis. Both of these alternatives are addressed in Annex A.

5.6 Isolated Receiver characteristics

For isolated devices, a capacitor shall be fitted between Pin 1 and chassis for the purpose of Radio Frequency bypass. Devices shall continue to operate correctly when exposed to any of the conditions in table 3.

Table 3 - Isolated Receiver characteristics

Connection	Limit	Comment
Pin 2 to Pin 1 or Pin 3 to Pin 1	+12 / -7 VDC	Common Mode range
Pin 4 to Pin 1 or Pin 5 to Pin 1	+12 / -7 VDC	Enhanced Function devices only
Pin 1 to Chassis	$\geq 22\text{M ohm @ } 42\text{ VDC}$	
Pin 2 to Chassis or Pin 3 to Chassis	$\geq 22\text{M ohm @ } 42\text{ VDC}$	
Pin 4 to Chassis or Pin 5 to Chassis	$\geq 22\text{M ohm @ } 42\text{ VDC}$	Enhanced Function devices only
Pin 2 to Pin 3	+/- 6 V	
Pin 4 to Pin 5	+/- 6 V	Enhanced Function devices only
Any Pin to Chassis	30 VAC / 42 VDC	

Figure 2 illustrates an isolated receiver. Any pin of the DMX512 isolated receiver shall present a resistance (B) greater than 22 Mohm at 42 VDC with respect to Chassis, with respect to Protective Ground (where fitted), with respect to any other signal inputs or outputs, and with respect to other ground referenced electronics. There shall be a capacitance (not shown) between Pin 1 and chassis for Radio Frequency bypass. Any resistance (A) between pin 1 and zero volt supply of the receiver circuit shall be less than 100 ohms.

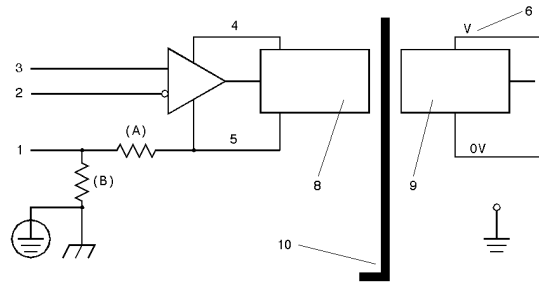


Figure Key

- 1 - DMX512 Pin 1
- 2 - DMX512 Pin 2 (or Pin 4)
- 3 - DMX512 Pin 3 (or Pin 5)
- 4 - Isolated Supply
- 5 - Isolated 0V
- 6 - V (+ or -)
- 7 - I / O
- 8 - Isolated Electronics
- 9 - Non-Isolated Electronics (optional)
- 10 - Isolation Barrier
- A - Optional Impedance (see text)
- B - Optional Impedance (see text)

Figure 2 - Isolated Receiver

Adherence to this topology allows a DMX512 receiver port to be marked as ISOLATED. Isolated receiver ports may be used by all DMX512 devices including ones that provide any number of non-DMX512 input or output ports. A DMX512 device may have any number of Isolated receiver ports.

5.7 Disallowed receiver topologies

This configuration is not permitted, although it may exist on some legacy products. While this topology is described as one possible topology in EIA-485-A, it is not appropriate when considering operation of DMX512 receiving devices in systems encountering differential ground potentials.

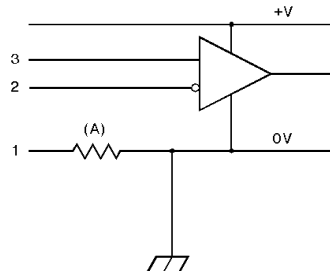


Figure Key

- 1 - DMX512 Pin 1
- 2 - DMX512 Pin 2 (or Pin 4)
- 3 - DMX512 Pin 3 (or Pin 5)
- A - $R \geq 0.2 \text{ ohm}$

Figure 3 - Receiver Topology NOT Allowed

5.8 Processing devices

It is also permissible to design processing devices based on the Isolated Receiver / Ground Referenced Transmitter or Isolated Receiver / Isolated Transmitter models (see Annex A) already described.

5.9 Loading designation

As per EIA-485-A, the total load permitted on a DMX512 data link is 32 unit loads. Transmitters designed for this Standard shall be capable of driving 32 unit loads on a DMX512 data link. All DMX512 devices shall have a unit load (LEN) of 1 or less.

A receiver biased to any voltage from -7 to +12 volts shall not present a capacitive load to the line of more than 200pf per unit load. If this value is frequency dependent, the value given shall be the capacitive load when driven by a 650kHz sine wave.

Declaring or marking of the unit load is not required by this standard. If a manufacturer chooses to declare or mark their products with a unit load value, the declared or marked value shall be the greater of either the DC unit load determined by EIA-485-A clause 4.1 or the unit load as determined by the capacitive loading. In either case, if the unit load is declared the capacitive load values must also be declared.

6 Protection

6.1 Minimum protection against interconnection damage

Clause 4.2 of EIA-485-A recognizes that certain other extraneous conditions may overstress the system and that these conditions should be specified in the referencing standard. Extensive use of temporary and portable equipment in Entertainment Lighting Industry results in frequent connection and disconnection of equipment and gives rise to the possibility of equipment misconnection.

Equipment may be protected against damage resulting from accidental connection to voltages in excess of the minimum defined in EIA-485-A clauses 4.2.5 and 4.2.6, and is recommended. See Annex C. This does not negate the need to comply with EIA-485-A clause 4.2.6 - Transient overvoltage tolerance.

6.2 Minimum Electro Static Discharge (ESD) protection

Manufacturers shall ensure that any pins can withstand a minimum of 4kV ESD for contact discharge and 8kV ESD for air discharge in accordance with IEC 61000-4-2 or other local regulations which may require higher levels of protection. The acceptance criteria for this requirement shall allow temporary loss of function, provided the function is self recoverable or can be restored by the operation of controls. Meeting this requirement does not alter the fact that a manufacturer may have to meet other more stringent ESD requirements to conform to local EMC regulations.

7 Connection Methods

7.1 Equipment fitted with external user accessible pluggable data link connections

This category includes all portable products.

Female connectors shall be used on controllers and other transmitting devices and male connectors shall be used on dimmers and other receiving devices. Female connectors shall also be used where loop-through connections are provided.

Marking and identification of all ports shall be as required in clause 11. Each DMX512 port with Enhanced Functionality shall be marked in accordance with clause 11 and Annex B.

7.1.1 Required Connector

Equipment in this category shall use 5-pin XLR connectors. The physical pinout of the 5-pin XLR shall be in accordance with the Pin reference in this Standard as defined in table 1.

7.1.2 Concession for use of an alternate connector (NCC DMX512-A)

A concession to use an alternate connector is available only when it is physically impossible to mount a 5-pin XLR connector on the product. In such cases the following additional requirements shall be met :

- 1) The alternate connector shall not be any type of XLR connector.
- 2) The alternate connector shall not be any type of RJ45 type connector except as allowed in clause 7.4.
- 3) Provided that all other requirements of this Standard are met, when marking is applied to any such alternate connector, it may be marked as NCC DMX512-A (Not Connector Compatible).
- 4) The manufacturer shall make available an adapter cable with the appropriate connections to a standard 5-pin XLR connector for all DMX512 ports included in the alternate connector.
- 5) The Enhanced Functionality, if applicable, and ground/isolation declarations shall continue to be declared for each port.

7.2 Equipment intended for fixed installation with internal connections to the data link

Products in this category may optionally use the 5-pin XLR connector, but shall not use any other XLR connector. Where a non 5-pin XLR connector is used, this Standard makes no other restriction or stipulation on connector choice.

When use is made of the 5-pin XLR connector, female connectors shall be used on controllers and other transmitting devices and male connectors shall be used on dimmers and other receiving devices. Female connectors shall also be used where loop-through connections are provided. In all other cases, the connector sex is not specified.

Products in this category with Enhanced Functionality shall be marked in accordance with the provisions of clause 11 and Annex B.

7.3 Passive Data Outlets or Wall plate panels

Products in this category designed for temporary and generally accessible access to DMX512 data links shall use 5-pin XLR connectors. Marking of such panels shall be in accordance with clause 11.5.4.

Panels which provide a data source on the primary data link shall use female connectors. Panels which are intended to provide primary data link input signals back to another location shall use male connectors.

7.4 RJ45 type connectors

Limited use of ISO IEC 11801 compliant cable schemes is permitted in accordance with clause 8.5 of this Standard. The use of RJ45 type connectors (plugs/jacks) and punchdown terminal blocks with this cable shall be limited to connections which are part of a fixed installation and not normally accessible or intended for regular connection and disconnection.

External (user accessible) RJ45 type connectors are permitted only on patch and data distribution products and only when installed in controlled access areas.

Products in this category with Enhanced Functionality shall be marked in accordance with the provisions of clause 11 and Annex B.

Installations using Category 5 cable implementing the one pair functionality of Cable Scheme C1 are referred to as Cable Scheme C5.1. This configuration limits systems to the basic functionality offered by TYPE 0 ports, but does permit two universes to be carried on one cable.

Installations using Category 5 cable implementing the two pair functionality of Cable Scheme C2 are referred to as Cable Scheme C5.2. The use of Cable Scheme C2 is required for the interconnection of products with EF1, EF2, or EF3 topologies, and is the preferred implementation.

Table 4 - Connection Schedule for DMX512 systems using ISO IEC 11801 Cable

Pair	Wire #	Color	Function	DMX512 Pin
Pair 2	1	white / orange	data 1 +	DMX512 Pin 3
	2	orange	data 1 -	DMX512 Pin 2
Pair 3	3	white / green	data 2+	DMX512 Pin 5
	6	green	data 2 -	DMX512 Pin 4
Pair 1	4	blue	Not assigned	
	5	white / blue	Not assigned	
Pair 4	7	white / brown	Signal Common (0 v)	DMX512 Pin 1
	8	brown	Signal Common (0 v)	DMX512 Pin 1
Shield		drain		

Note: ISO IEC 11801 cable wire pair numbering and color in accordance with TIA T568B.

Warning: Accidental connection to non-DMX512 equipment likely to be encountered (e.g., an Ethernet Hub at a patch bay) may result in damage to equipment.

Warning: Wires 4 and 7 are used for various purposes in other wiring standards, including telephone ringing voltage. Some manufacturers whose distributed DMX512 buffering products require low voltage DC power may use these wires for this purpose. Because of these various uses, misplugging unlike systems could cause serious damage. However, since these wires are never connected to an end user XLR connector, mitigation of potential problems involving such systems are currently beyond the scope of this Standard.

8 Cable

8.1 Background

The data transmission rate (250 kbits/s) used by DMX512 requires the selection of a cable which does not significantly distort the signal or give rise to spurious signal reflections. Cables intended for use with audio systems (microphone cables), while having the convenience of flexibility, availability and relative low cost, are **NOT** suitable for use with DMX512 because of their high capacitance and incorrect characteristic impedance; at DMX512 data rates this will give rise to bit time distortion and signal reflections/overshoot, particularly over long (greater than 10 meter) distances.

8.2 General

Cabling systems shall provide a balanced, nominal 120 ohm terminated transmission system, and be made of cables with a characteristic impedance in the range 100-120 ohm.

Note: Note that mixing cables of different impedances and other characteristics not isolated by buffers or other processing devices may affect system reliability.

Cables with one pair are referred to as Cable Scheme C1 in this Standard. The use of Cable Scheme C1 does not allow for Enhanced Functionality and is restricted to portable cables only. Enhanced Functionality product may be interconnected but operation will be limited to transfer of data on the primary data link.

Cables with two pairs are referred to as Cable Scheme C2 in this Standard. The use of Cable Scheme C2 is required for permanent wiring and for portable cables used for the interconnection of products with Enhanced Functionality.

8.3 General Applications and between all Portable Equipment

Cable for general application shall be shielded twisted pair approved by its manufacturer for EIA-422/EIA-485-A use at high data transmission rates and distances of at least 500 meters. Conductors in cable for portable equipment shall be of stranded construction.

8.4 Cable between permanently installed fixed equipment

Any two-pair cable satisfying the requirements of clauses 8.2 and 8.3 may be used for connections between items of fixed equipment, subject to local regulatory requirements regarding voltage and insulation styles.

The use of shielded or unshielded 100 ohm or 120 ohm ISO IEC 11801 compliant cable shall be permitted.

Conductors in cable installed permanently between fixed equipment shall be permitted to be solid.

8.5 Cable application rules

Table 4 summarizes the various constructions of cable suitable for use with DMX512 and how they can be implemented.

Table 5 - Cable application rules

Use	Solid Conductors			Stranded Conductors		
	EIA-485-A	UTP 100 or 120 ohm ISO IEC 11801	STP/FTP 100 or 120 ohm ISO IEC 11801	EIA-485-A	UTP 100 or 120 ohm ISO IEC 11801	STP/FTP 100 or 120 ohm ISO IEC 11801
Portable	No	No	No	OK - Note 1 -	No	OK - Note 1 -
Permanent	OK	In Earthed Metal Conduit Only	OK - Note 2 -	OK - Note 2 -	In Earthed Metal Conduit Only	OK - Note 2 -

Note 1: It is recommended that braided shield be used for better durability

Note 2: Use of Plenum Rated cable without conduit allowed

9 Data Protocol

9.1 Format

Data transmitted shall be in asynchronous serial format. DMX512 slots shall be transmitted sequentially, beginning with slot 0 and ending with the last implemented slot, up to a maximum of 513. Prior to the first data slot being transmitted, a Reset Sequence shall be transmitted – a BREAK, followed by a MARK AFTER BREAK, and a START Code. Valid DMX512 slot values under a NULL START Code shall be 0 to 255 decimal.

9.2 Slot format

The data transmission format for each data value transmitted shall be as follows:

Table 6 - Data slot format

Bit Position	Description
1	Start Bit, Low or SPACE
2 through 9	Slot Value Data Bits, Least Significant Bit to Most Significant Bit Positive logic
10, 11	Stop Bits, High or MARK
Parity	Not transmitted

9.3 Break

The BREAK (Timing Diagram, Designation #1) shall be defined as a high-to-low transition followed by a low of 88 microseconds (two slot times) duration or longer followed by a low to high transition. The BREAK indicates the start of a new packet.

9.4 Mark After Break

The duration of the MARK separating the BREAK and the START Code (Timing Diagram, Designation #2) shall be not less than 8 microseconds and not greater than 1 second. All DMX512 transmitters shall produce a MARK AFTER BREAK of not less than 8 microseconds. All receivers shall recognize an 8 microsecond MARK AFTER BREAK.

Note: The 1986 version of this standard specified a 4 microsecond MARK AFTER BREAK. The 1990 version of the standard changed that value to 8 microseconds, but added an option for receivers capable of recognizing the 4 microsecond MARK AFTER BREAK to be identified as having that capability. Some transmitters may still be in use that generate the shorter 4 microsecond MARK AFTER BREAK, and they may not work with equipment built to this standard.

9.5 START Code

The START Code is the first byte following a MARK AFTER BREAK. The START Code identifies the function of subsequent data in that packet.

9.5.1 NULL START Code

The NULL START Code (a NULL byte – all zeros) identifies subsequent data as sequential 8-bit information.

9.5.2 Other START Codes

In order to provide for future expansion and flexibility, DMX512 makes provision for 255 additional non NULL START Codes (1 through 255 decimal, 01 through FF hexadecimal), henceforth referred to as Alternate START Codes. Where it is required to send proprietary information over a DMX512 data link, a packet starting with a registered Alternate START Code shall be used.

Annex D states the requirements for transmitting, processing, and receiving Alternate START Codes.

Several Alternate START Codes are reserved. See Annex E.

See Annex F for Alternate START Code Registration Policies.

9.5.3 START Code processing

All receiving devices other than in-line processing devices shall process the START Code and differentiate between those packets with NULL START Codes and those with Alternate START Codes. Devices shall not ignore START Codes by assuming that all packets received are NULL START Code packets.

9.6 Maximum number of data slots

Each data link shall support up to 512 data slots. Multiple links shall be used where larger numbers of slots are required.

9.7 Minimum number of data slots

There shall be no minimum number of data slots on the data link. DMX512 data packets with fewer than 512 slots may be transmitted, subject to the minimum timing requirements of this standard – see clause 9.10 and figure 4.

9.8 Defined line state between slots

The time between any two slots of a data packet (Timing Diagram, Designation #9) may vary between 0 microseconds and 1 second. The line must remain in a "marking" state during any such idle period. A receiver must be capable of accepting a data packet having no idle time (0 microseconds) between any of its slots.

9.9 Defined line state between data packets (Mark Before Break)

Every data packet transmitted on the data link, regardless of START Code or length, must begin with a BREAK, MARK AFTER BREAK, and START Code sequence as defined above. The time between the second stop bit of the last data slot of one data packet and the falling edge of the beginning of the BREAK for the next data packet (Timing Diagram, Designation #10) may vary between 0 microseconds and 1 second. The line shall remain in an idle ("marking") state throughout any such period greater than 0 microseconds. Transmitters, therefore, shall not produce multiple BREAKs between data packets. Receivers, however, shall be capable of recovering from multiple BREAKs produced by data link line errors.

9.10 Break-to-Break spacing

The period between the falling edge at the start of any one BREAK shall be not less than 1196 microseconds from the falling edge at the start of the next BREAK, nor more than 1.025 seconds.

9.11 Dimmer class data

Valid dimmer levels shall be 0 to 255 decimal (00 to FF hexadecimal) representing dimmer control input. 0 shall represent a dimmer output of OFF or minimum and 255 shall represent an output of FULL. A dimmer shall respond to increasing the DMX512 slot value for 0 to 255 by fading from its minimum level (off) to its maximum level (full). The exact relationship between DMX512 slot values and dimmer output is beyond the scope of this Standard.

9.12 Timing Diagram - output of transmitting UART

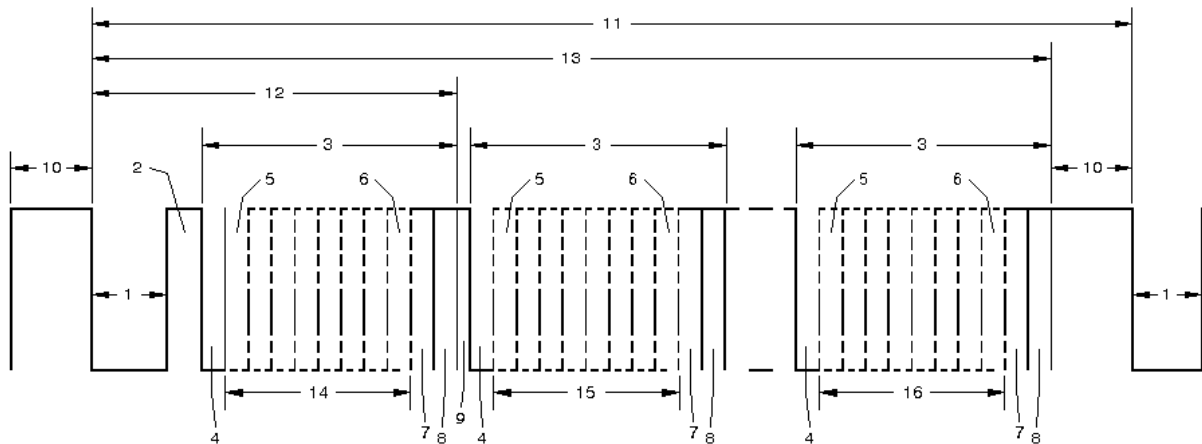


Figure Key

- 1 - "SPACE" for BREAK
- 2 - "MARK" After BREAK (MAB)
- 3 - Slot Time
- 4 - START Bit
- 5 - LEAST SIGNIFICANT Data BIT
- 6 - MOST SIGNIFICANT Data BIT
- 7 - STOP Bit
- 8 - STOP Bit
- 9 - "MARK" Time Between slots
- 10 - "MARK" Before BREAK (MBB)
- 11 - BREAK to BREAK Time
- 12 - RESET Sequence (BREAK, MAB, START Code)
- 13 - DMX512 Packet
- 14 - START CODE (Slot 0 Data)
- 15 - SLOT 1 DATA
- 16 - SLOT nnn DATA (Maximum 512)

Figure 4 - Timing Diagram

Table 7 - Timing Diagram Values

Designation	Description	Min	Typical	Max	Unit
-	Bit Rate	245	250	255	kbit / s
-	Bit Time	3.92	4	4.08	µs
-	Minimum Update Time for 513 slots	–	22.7	–	ms
-	Maximum Update Rate for 513 slots	–	44	–	/ s
1	“SPACE” for BREAK	88	–	–	µs
2	“MARK” After BREAK (MAB)	8	–	< 1.00	µs s
9	“MARK” Time between slots	0	–	< 1.00	s
10	“MARK” Before BREAK (MBB)	0	–	< 1.00	s
11	BREAK to BREAK Time	1196	–	– 1.00	µs s
13	DMX512 Packet	1196	–	– 1.00	µs s

10 Receiver Performance

10.1 Loss of data tolerance / Resumption of acceptance of data

A receiver not receiving a Reset Sequence (a sequence of a Break, Mark After Break, and START Code) within one second of the previous Reset Sequence shall be considered to have lost data input.

Although this Standard does not specify loss of data handling procedures, manufacturers shall state what their Loss of Data handling procedures are.

Note: In the absence of overriding safety issues or an alternative source of control data, when encountering a loss of data condition a receiving device should remain in an operating condition for at least 60 seconds, awaiting resumption of the DMX512 signal.

10.2 Receiver performance at maximum update rate

Any device incorporating a DMX512 receiver shall operate correctly when receiving continuous transmission of valid data packets containing any number of slot values.

10.3 Inactive receiver input circuitry

Unpowered connected DMX512 devices shall not degrade the performance of the DMX512 transmission system.

10.4 Packet processing latency

Some products may provide their specified functionality without processing or being able to process every consecutive DMX512 packet. Such products will have an inherent latency to data changes between packets, which shall be declared by the manufacturer in accordance with the disclosure requirements of clause 11.

11 Marking and Disclosures

11.1 Identification

Only equipment conforming to this Standard may be marked and identified with "USITT DMX512-A" or "DMX512-A".

11.2 Port marking

Where required by clauses 11.3 through 11.4, all ports shall be marked as to the applicable Enhanced Functionality as defined in table B1 (Annex B). All information provided by marking shall also be provided in the equipment manual.

Where declaration of pinout detail is required by clauses 11.3 through 11.4, manufacturers shall use the signal designations from table 1 in any marking of pin, contact or terminal functions appearing on or within a product and associated with installation or connection. Such details are dependent on the declared Enhanced Functionality of the port. Where it is necessary to use abbreviations, only those detailed in table 8 shall be permitted.

Table 8 - Signal designations abbreviations allowed for marking

Function	Abbreviation
Common (Screen)	COM
Primary Data Link – Data 1-	D1-
Primary Data Link – Data 1+	D1+
Secondary Data Link – Data 2-	D2-
Secondary Data Link – Data 2+	D2+


11.3 Data line termination marking

On portable products and products fitted with external pluggable data link connectors, clear and appropriate identification of the termination state shall be provided when the port termination is switchable.

11.4 Ground / Isolation marking

All DMX512 ports shall also be marked as to the relationship between Pin 1 and earth ground. The allowed grounding topologies are shown in table 9.

Table 9 - Ground / Isolation marking

Function of Port	Defining Figure	Comment	Approved Marking
Transmitter	fig 1	Ground Referenced	<i>no mark required</i>
Transmitter	fig A1	Isolated (See Annex A)	ISO or ISOLATED
Transmitter	fig A4	Floating (See Annex A)	FLT or FLOAT or FLOATING
Receiver	fig A2	Non-Isolated, 100 ohm 2 Watt resistor	NON-ISO or NON-ISOLATED
Receiver	fig A3	Grounded – concession per note 3 of the figures (See Annex A)	 PIN 1 \perp
Receiver	fig 2	Isolated	<i>no mark required</i> ; ISO or ISOLATED
Receiver	fig A4	Floating	FLT or FLOAT or FLOATING

11.5 Required disclosures

11.5.1 Portable products and products fitted with external pluggable data link connectors

Ports on these products shall be marked in accordance with clause 11.2. Ports on these products shall provide Ground/Isolation marking in accordance with clause 11.4.

If use has been made on any non-XLR connector in conjunction with the supply of an adapter cable, the non-XLR connector may be also be marked as NCC DMX512-A.

11.5.2 Equipment intended for fixed installation with internal connections to the data link

Ports on these products shall be marked in accordance with clause 11.2. Ports on these products shall provide Ground/Isolation marking in accordance with clause 11.4.

Clearly identified connector pinout detail shall be marked on or within any product in accordance with clause 11.2.

11.5.3 Internal connections and specialized products for use with ISO IEC 11801 cable schemes

Ports on these products shall be marked in accordance with clause 11.2. Ports on these products shall provide Ground/Isolation marking in accordance with clause 11.4.

11.5.4 Data outlet or wall plates fitted with 5-pin XLR connectors

Where a 5-pin XLR female or male connector is fitted to a wall plate or facility panel in a manner that links it directly back to a patch panel or data distribution buffer, its Enhanced Functionality, if it exists, cannot be determined without reference to the product to which it is ultimately patched and the cable scheme implemented. Therefore, manufacturers and installers of such data outlets shall be permitted to mark the XLR connectors in accordance with table 10. If the site cable scheme or pinout does not follow this table, no reference to DMX512, including use of the term “DMX”, shall be permitted.

No reference to DMX512-A shall be permitted, since the “A” indicates electrical characteristics of active electronics compliant with this edition of the Standard, which cannot always be known at the passive wall plate.

Table 10 - 5-pin XLR data outlet / wall plate marking

5-pin XLR - Pins Wired	Cable Scheme	Permitted Marking at XLR (the word “Scheme” is optional)	Comments
1, 2, 3	C1	DMX512 Scheme C1	Not allowed for new installations
1, 2, 3, 4, 5	C2	DMX512 Scheme C2	Required for new Installations

11.5.5 Loss of data handling procedure

This Standard does not define loss of data handling procedures (clause 8.1) beyond requiring that manufacturers declare their own products’ procedure(s). Such declarations shall be made in the equipment manual.

11.5.6 Packet Processing latency

Manufacturers shall declare any inherent latency to data changes between packets in the equipment manual. This may be in terms of response time or other wording as chosen by the manufacturer, but shall clearly indicate if the product design might legitimately ignore some packets in normal operation.

11.5.7 NULL START Code functionality

Manufacturers of transmitting devices shall declare in the device manual the full range of slot values transmitted in conjunction with packets sent using the NULL START Code.

Manufacturers of receiving devices shall declare the response to packets received containing the NULL START Code, with particular reference to any functionality requiring limited or restricted slot data values, in the equipment manual.

11.5.8 Slot footprint

Manufacturers of receiving devices shall declare the slot footprint in the equipment manual.

= END =

Annex A - Alternate (not preferred) topologies (Normative)

A1 Isolated transmitters

This standard permits the use of isolated transmitter ports. They are used in systems where legacy receivers or grounded receivers are used and large common mode voltages are expected.

Figure A1 illustrates an isolated transmitter port. To be considered an isolated transmitter, any pin of the DMX512 output shall present a resistance (B) greater than 22 Mohm at 42 VDC with respect to Chassis, and with respect to Protective Ground (where fitted). If there are any other signal inputs or outputs any pin of the DMX512 output shall present a resistance greater than 22 Mohm at 42 VDC with respect to these inputs or outputs. The power supply for the transmitter and any directly connected electronics shall be isolated from earth ground and any other grounded referenced electronics to levels at least as high as those required of the output pins. Any impedance (A) between pin 1 and zero volt supply of the transmitter circuit shall be less than 100 ohms. There shall be a capacitance (not shown) between Pin 1 and chassis for the purpose of Radio Frequency bypass.

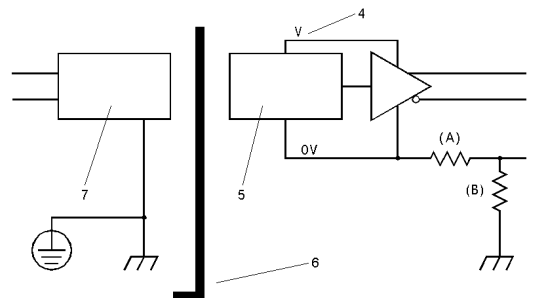


Figure Key

- 1 - DMX512 Pin 1
- 2 - DMX512 Pin 2 (or Pin 4)
- 3 - DMX512 Pin 3 (or Pin 5)
- 4 - V (+ or -)
- 5 - Other Isolated Electronics
- 6 - Isolation Barrier
- 7 - Non-Isolated Electronics (optional)
- A - Optional Impedance (see text)
- B - Impedance (see text)

Figure A1 - Isolated Transmitter

Table A1 - Isolated Transmitter characteristics

Connection	Limit	Comment
Pin 2 to Pin 1 or Pin 3 to Pin 1	+/- 6 V	Common Mode range
Pin 4 to Pin 1 or Pin 5 to Pin 1	+/- 6 V	Enhanced Function devices only
Pin 1 to Chassis	$\geq 22\text{M ohm @ 42 VDC}$	
Pin 2 to Chassis or Pin 3 to Chassis	$\geq 22\text{M ohm @ 42 VDC}$	
Pin 4 to Chassis or Pin 5 to Chassis	$\geq 22\text{M ohm @ 42 VDC}$	Enhanced Function devices only
Pin 2 to Pin 3	+/- 6 V	
Pin 4 to Pin 5	+/- 6 V	Enhanced Function devices only
Any Pin to Chassis	30 VAC / 42 VDC	

A DMX512 device may have any number of isolated transmitter ports.

Adherence to this topology allows a DMX512 transmitter connector to be marked ISO or ISOLATED.

A2 Non-isolated receivers

While not the preferred topology, this non-isolated topology exhibits considerable improvement in common tolerance compared to other grounded or non-isolated topologies. In this topology there shall be a resistance (B) of 100 ohms between pin 1 and chassis. This resistance shall be able to safely dissipate two watts. No other connection between circuit common and chassis is permitted. Common mode voltage present between the chassis of this device and the chassis of any other device connected to the DMX shield will cause a current flow. This current will cause a voltage drop in the resistance (B). This voltage drop effectively decreases the common mode voltage at this receiver. Any resistance (A) between pin 1 and zero volt supply of the receiver circuit shall be less than 100 ohms.

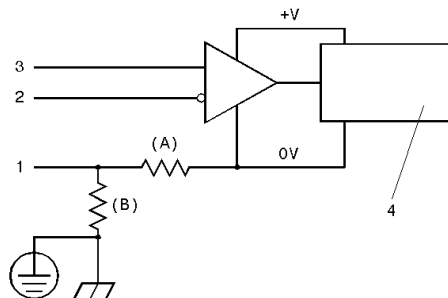


Figure Key

- 1 - DMX512 Pin 1
- 2 - DMX512 Pin 2 (or Pin 4)
- 3 - DMX512 Pin 3 (or Pin 5)
- 4 - Other Electronics
- A - Optional Impedance (see text)
- B - Optional Impedance (see text)

Figure A2 - Non-Isolated Receiver

Since 0 V is not directly referenced to Chassis, local product safety standards may restrict choice of power supply (e.g., use of a Class 2 supply).

A DMX512 receiver may have any number of non-isolated receiver ports. Where multiple ports are implemented, the total parallel resistance (B) shall be 100 ohms.

Table A2 - Non-Isolated Receiver characteristics

Connection	Limit	Comment
Pin 2 to Pin 1 or Pin 3 to Pin 1	+12 / -7 VDC	Common Mode range
Pin 4 to Pin 1 or Pin 5 to Pin 1	+12 / -7 VDC	Enhanced Function devices only
Pin 1 to Chassis	100 ohms	-Note 1-
Pin 2 to Chassis or Pin 3 to Chassis	+12 / -7 VDC	
Pin 4 to Chassis or Pin 5 to Chassis	+12 / -7 VDC	Enhanced Function devices only
Pin 2 to Pin 3	+/- 6V	
Pin 4 to Pin 5	+/- 6V	Enhanced Function devices only
Any Pin to Chassis	N/A	

Note 1 : This cannot be characterized in terms of voltage. Manufacturers shall be permitted to fit a resistance of 100 ohms +/-20% between Chassis and Pin 1 for the purpose of limiting current in the screen due to small differential ground potentials. This method provides for reduction of Common Mode voltage at the line receiver.

The only output permitted to be directly connected to this topology is a single passive DMX512 loop through port. No other inputs or outputs are allowed with this topology unless they meet the requirements of clause A4.

A3 Grounded Receivers

he topology of figure A3 is allowed for the construction of entry level receivers where the cost of isolation might prove an untenable burden. It may be used by manufacturers of receivers who, for reasons beyond the scope of this Standard require a direct link between data link common and protective earth. It is not a recommended practice and requires special marking and special explanatory text in all manuals.

This topology is characterized by the direct connection of the shield (Pin 1) to chassis and protective earth. Therefore, devices employing ground referenced receivers shall be provided with provision for connection to protective earth. Any resistance (A) between pin 1 and zero volt supply of the transmitter circuit shall be less than 100 ohms.

The only DMX512 output permitted to be directly connected to this topology is a single passive DMX512 loop through port.

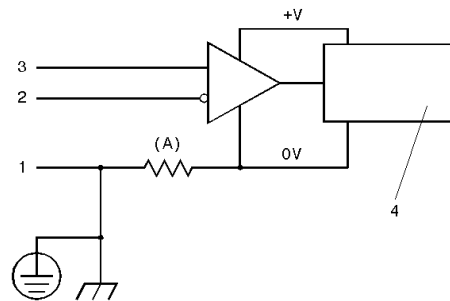


Figure Key

- 1 - DMX512 Pin 1
- 2 - DMX512 Pin 2 (or Pin 4)
- 3 - DMX512 Pin 3 (or Pin 5)
- 4 - Other Electronics
- A - Optional Impedance (see text)
- B - Optional Impedance (see text)

Figure A3 - Grounded Receiver

Ports using this topology shall be marked using the standard symbol representation as defined in clause 11.4 and be declared.

A4 Earth grounding of data link common for floating devices

Figure A4 illustrates a floating topology. Floating is both an input and an output topology and is an additional allowable topology. It is often confused with the isolated DMX topology. As with the isolated topology any input or output pin shall present a resistance (B) greater than 22 Mohm at 42 VDC with respect to Chassis, and with respect to Protective Ground (where fitted). It differs from an isolated topology in that there is no requirement of isolation between DMX512 inputs and outputs. The resistance between pin 1 of the DMX512 input and pin 1 of the DMX512 output shall be less than 0.2 ohms. There shall be a capacitance (not shown) between Pin 1 and chassis for the purpose of Radio Frequency bypass. Any resistance(B) between any pin 1 and the zero volt supply of the transmitter and receiver circuits shall be less than 100 ohms.

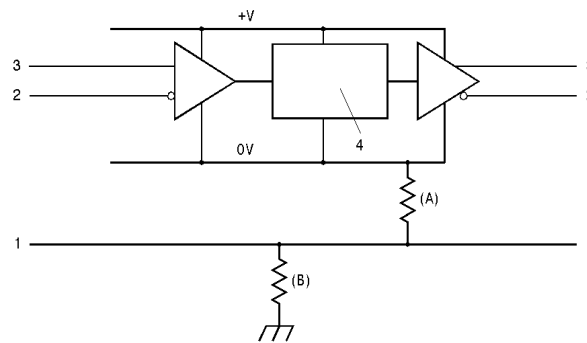


Figure Key

- 1 - DMX512 Pin 1 (Data Link Common)
- 2 - DMX512 Pin 2 (or Pin 4)
- 3 - DMX512 Pin 3 (or Pin 5)
- 4 - Other Electronics
- A - Optional Impedance (see text)
- B - Optional Impedance (see text)

Figure A4 - DMX512 Device, Floating

The grounding of a device using this topology is determined by the connected devices. For that reason this topology shall not be used for devices that provide ground referenced non-DMX512 input or output ports unless those ports are isolated from DMX512 lines by an impedance of at least 22 Mohms.

Both input and output ports of a floating device shall be marked FLT, FLOAT, or FLOATING. A device may have only one floating DMX512 receiver port.

-end of Annex A-

This Page Almost Blank

Annex B (Normative) - Enhanced DMX512, the second data link

B1 General

The original and 1990 versions of USITT DMX512 called out an “Optional Second Data Link.” There was no detailed guidance for its use. The majority of legacy systems did not use the second data pair at all. Many uses of the second data pair have been implemented over the years. While many of these were reasonable, a few uses clearly were not compliant with EIA-485-A. These uses vary in both their electrical requirements and in the data protocol used. One of the purposes of this standard is to regularize the use of the second data pair. It is no longer possible to select a single implementation and forbid all others. However, not all historical uses will be allowed to continue. Others are allowed so that manufacturers may support installed legacy equipment, will not be recommended for new designs. New uses of the second data link will be permitted when it is determined that they offer general benefit to the end user while not adversely affecting interoperability of existing compliant equipment.

The network topologies needed to support Enhanced Functions are identified by an EF number. This edition of the standard supports three different EF topologies.

The introduction of standard EF topologies in no way changes backward compatibility of primary data link functions. In all cases, regardless of the EF topology, all DMX512 transmitters and receivers shall be able to interchange DMX512 data on the primary data link. All DMX512 devices shall be able to be connected without damage. All DMX512 devices shall not be damaged by connection to compliant legacy devices.

B2 Summary of Enhanced Function Topologies

The Enhanced Function topologies are summarized in table B1. The difference between basic DMX512-A and EF1 or EF2 is the signal topology of the secondary data link only. EF3 allows for the limited use of the primary data link in a bi-directional mode.

Table B1 - Enhanced Function topologies

EF #	Symbol	Description	Comment
1	→ ←	unidirectional DMX512 data Pins 2, 3 (EIA-485-A signals) and return EIA-485-A signals on Pins 4, 5	
2	→ ↔	unidirectional DMX512 data Pins 2, 3 (EIA-485-A signals) and half duplex EIA-485-A signals on Pins 4, 5	refer to manufacturers instructions
3	↔ ↔	half duplex EIA-485-A signals on both pairs; return signal on Pins 2, 3 controlled by a registered Alternate START Code	the data format on 2 & 3 must be compatible with standard DMX512

Note: References to unidirectional data are with respect to a transmitting device.

Different physical ports on a product may be of different EF topologies and must be declared and marked as required in clause 11.

B3 Identification of data protocols on the secondary data link

The EF topology specifies the electrical and operating mode for the data links. It does not detail the data structure or protocol. The method of identifying data structures or protocols using the defined EF numbers is by way of an additional registered number placed after the EF number, separated from the EF number by a period. The registry for these designations will be maintained by Secretariat for the ANSI E1 Accredited Standards Committee – ESTA (see Annexes E and F). Numbers ending in zero are reserved for future development of the standard (e.g., EF1.0, EF1.10, EF2.20).

B3.1 EF1- Full duplex DMX512

The EF1 topology uses the second data link to provide a data path to return status information from controlled devices. The data on second data link shall be unidirectional flowing from the controlled devices to the controller. This data link will operate in multi transmitter multipoint mode. In this mode the primary and secondary links comprise a full duplex data link. A DMX512 controlled device capable of returning status information is referred to as a responder in this standard.

Each responder shall have an EIA-485-A transmitter for the second data link with a driver enable control. The driver enable control shall be driven so that each responder can control the state of the return data link while transmitting its response bytes. Once configured only one responder shall be enabled at once and there shall be at least one bit time between one responder going inactive and the next going active. In general, the line driver can be enabled one bit time prior to transmission, and disabled one bit time after the last bit has been sent.

Once configured, any functions using EF1 shall use collision free data protocols. EF1 protocols shall also be structured to allow the use of data distribution amplifiers that meet the requirements of clause B3.1.3.

B3.1.1 Wiring of EF1 ports

EF1 responders having two ports for receive and transmit must provide a direct passive link on Pins 4 and 5 between those ports (clause 4.8.2). The responder transmitter shall be connected to this passive link. Devices having three or more ports shall wire the additional ports in a manner appropriate for the device's functionality. The manner in which these ports are wired shall be clearly detailed in the product's manual. Devices having three or more ports shall have two ports wired as a loop through with an attached responder transmitter.

B3.1.2 Bi-directional distribution amplifiers

Systems using EF1 topologies require bi-directional distribution/return data combiners. The DMX512 primary data pair data may be split and separately buffered as in standard DMX512 buffers. Return data receivers shall be "wire-OR" connected within a unit. This combined received data signal is used to drive back to the return data monitor.

The bi-directional distribution amplifier shall not perform any processing on the data, since some talkback protocols depend upon the relationship with outbound DMX512 to synchronize the return data.

B3.2 EF2 Half duplex on the second data pair

Systems that send data in both directions on the secondary data link are classified as EF2. These systems shall use the primary data link to send a standard unidirectional DMX512 signal.

B3.3 EF3 Protocols that use the primary and the secondary data links in ways not covered above

Systems that send data in both directions on the primary data link are classified as EF3. These systems shall use the primary data link for both the Null START code DMX512 signal packets as well as return data controlled by the use of Alternate START Code packets. EF3 is not recommended for new implementations.

In an EF3 system, use of the secondary data link is optional. However, all devices that are primarily receivers or that act as processing devices shall support the use of half-duplex bi-directional communication on the secondary pair.

An EF3 system shall use polled feedback on the primary data link, such that a receiving device can not transmit a packet unless instructed to by an appropriate Alternate START Code packet from the controller. All response packets shall begin with an Alternate START Code.

All packets in an EF3 system shall follow the normal timing and rules for a DMX512 packet.

A response message to an ASC packet sent from the controller shall not be sent less than 90uS after receiving the command from the controller. This is to allow for turnaround time in the transceiver. The controller shall allow up to 2.5mS for the response message to be received. After that time, the controller can assume the response to be lost, and resume NSC packets or send another ASC request packet.

A receiver in transmit mode sending a response message shall release the line back to the controller within 44uS of completing its packet.

-end of Annex B-

This Page Almost Blank

Annex C (Informative) - Higher Protection Levels

Some manufactures feel it is prudent to employ higher levels of protection on their DMX512 ports than is specified in EIA-485-A. Many semiconductor manufacturers have recognized the need for higher protection and have produced “fault protected transceivers.” The tables below indicate the minimum requirements for a protected DMX512 port. If manufacturer’s DMX512 port meets these requirements, the port can be declared as protected.

Table C1 - Transmitter protection

Connection	Minimum Protection Limit			Comment
Pin 2 to Pin 1 or Pin 3 to Pin 1	30 VAC / \pm 42 VDC			
Pin 4 to Pin 1 or Pin 5 to Pin 1	30 VAC / \pm 42 VDC			
Pin 2 to Pin 3	30 VAC / \pm 42 VDC			
Pin 4 to Pin 5	30 VAC / \pm 42 VDC			
	Ground Referenced	Isolated	Floating (Note 1)	
Pin 1 to Chassis	shall not exceed 0.2 ohms	N/A	N/A	See clause 6.3
Any other Pin to Chassis	30 VAC / \pm 42 VDC	N/A	N/A	
Any Pin to Chassis	N/A	30 VAC / \pm 42 VDC	30 VAC / \pm 42 VDC	

Note 1 : Floating heading to indicate that this is only encountered on DMX512 Processing equipment.

Table C2 - Receiver protection

Connection	Minimum Protection Limit			Comment
Pin 2 to Pin 1 or Pin 3 to Pin 1	30 VAC / \pm 42 VDC			
Pin 4 to Pin 1 or Pin 5 to Pin 1	30 VAC / \pm 42 VDC			
Pin 2 to Pin 3	30 VAC / \pm 42 VDC			
Pin 4 to Pin 5	30 VAC / \pm 42 VDC			
	Non-Isolated	Isolated	Floating (Note 2)	
Pin 1 to Chassis	-note 1-	N/A	N/A	
Any other Pin to Chassis	30 VAC / \pm 42 VDC	N/A	N/A	
Any Pin to Chassis	N/A	30 VAC / \pm 42 VDC 100Mohm at 50 VDC	30 VAC / \pm 42 VDC \geq 22Mohm at 50 VDC	

Note 1: This cannot be characterized in terms of voltage. Clause A2 (Receiver Characteristics) allows manufacturers to fit a resistance between chassis and Pin 1 for the purpose of limiting the current in the screen (shield) due to small ground differentials. Any such resistance shall survive continuous connection to voltages within the EIA-485 Common Mode range of -7/+12 VDC. Manufacturers shall ensure that any failure of this component due to exposure to voltages not exceeding 30 VAC / 42 VDC will not cause a safety hazard.

Note 2: Floating heading to indicate that this is only encountered on DMX512 Processing equipment.

This Page Almost Blank

Annex D (Normative) - Alternate START Codes Implementation

D1 ASC refresh interval

A DMX512 transmitter interleaving NULL START Code packets with Alternate START Code packets shall send a NULL START Code packet at least once per second.

D2 Timing differences for Alternate START Code packets

To ensure that in-line processing devices do not lose essential Alternate START Code data, a reduction in the Alternate START Code update rate may be necessary. Recommended methods of achieving this are:

1) Alternate START Code packets may be transmitted slower than the minimum timings specified in clause 9.12 by increasing the break to break time to 10% more than the minimum required for the Alternate START Code packets number of slots.

2) Transmitters may interlace non essential Null START Code packets with Alternate START Code packets.

D3 Handling of Alternate START Code packets by in-line devices

DMX512 processing devices or any device that receives and re-transmits DMX512 shall state in the manual for the product how they process Alternate START Code packets. The acceptable processing methods are:

1) Block all packets containing particular Alternate START Codes. The START Codes blocked must be declared (and may be all Alternate START Codes).

2) Pass unmodified all packets containing particular Alternate START Codes. The START Codes passed must be declared.

3) Process the information contained in packets containing particular Alternate START Codes. The algorithm must be stated in enough detail to allow the user to decide if the device will satisfy their needs.

DMX512 in-line repeating transmitters should not pass some packets with a particular Alternate START Code while blocking other packets containing the same Alternate START Code unless doing so as part of a stated processing algorithm.

This Page Almost Blank

Annex E (Normative) - Reserved Alternate START Codes

E1 Reserved Alternate START Codes

Several Alternate START Codes are reserved for special purposes or for future development of the Standard.

Table E1 - Reserved Alternate START Codes

Alternate START Code		Purpose	Note
Hexadecimal	Decimal		
17	23	Text Packet	see Annex Clause E2 for implementation
55	85	Test Packet	see Annex Clause E3 for implementation
90	144	90h is reserved for future expansion	
91	145	91h followed by a 2 byte Manufacturer ID field is reserved for Manufacturer/Organization specific use – the first byte after the Manufacturers ID would normally be a manufacturer's sub-code	The Manufacturer ID serves as an identifier that the data following in that packet is proprietary to that entity and should be ignored by all others
92 - A9	146 - 169	possible future revisions of this Standard	No equipment shall be manufactured that generates any of these codes until their use is defined by the Standard
AB - CD	171 - 205	possible future revisions of this Standard	No equipment shall be manufactured that generates any of these codes until their use is defined by the Standard
CF	207	System Information Packet	see Annex E4 for implementation
F0 - F7	240 - 247	prototyping/experimental use while the manufacturer/organization is waiting for their registered Alternate START Code to be assigned	Manufacturers shall not advertise or sell products or devices that use Alternate START Codes F0 - F7.

E2 ASC text packet

Alternate START Code 17h (23 decimal) shall designate a special packet of between 3 and 512 data slots. The purpose of the ASC text packet is to allow equipment to send diagnostic information formatted for display.

Slot allocation is as follows:

Slot 1: Page number of one of the possible 256 text pages.

Slot 2: Characters per Line. Indicates the number of characters per line that the transmitting device has used for the purposes of formatting the text. A slot value of zero indicates ignore this field.

Slots 3-512: Consecutive display characters in ASCII format. All characters are allowed and where a DMX512 text viewer is capable, it shall display the data using the ISO/IEC 646 standard character set. A slot value of zero shall terminate the ASCII string.

E3 ASC test packet

Alternate START Code 55h (85 decimal) shall designate a special test packet of 512 data slots, where all data slots carry the value 55h (85 decimal). Test packets shall be sent so that the time from the start of the Break until the stop bit of the 513th slot shall be no more than 25 milliseconds. When test packets are sent back to back, the Mark Before Break time shall be no more than 88 microseconds. The Break timing for test packets shall be greater than or equal to 88 microseconds, and less than or equal to 120 microseconds. The Mark After Break time shall be greater than or equal to 8 microseconds and less than or equal to 16 microseconds.

E4 System Information Packet (SIP) Alternate START Code

Alternate START Code CFh (207 decimal) is reserved for a System Information Packet (SIP). The SIP includes a method of sending checksum data relating to the previous packet on the data link and other control information.

E4.1 Application

Manufacturers of control consoles are encouraged to transmit SIPs, either as a background to normal processing or, in conjunction with the special test packet, as part of their suite of system test functions. One of the current problems with testing of DMX512 installations is that it must be done with static test packets – certain modes of testers cannot be used while a console is actually running the show, as by definition the DMX512 packets are varying as each cue runs. The interleaving of SIP's would allow some degree of live testing, particularly if one of more test packets were also sent applicable to the functionality of the receiving device.

Note: For systems requiring a more reliable link, manufacturers would have the option of following every normal packet with a SIP packet, although it is recognized that this would degrade data throughput. It could be used with systems that send packets of fewer than 512 DMX512 data slots or refresh data at less than the maximum rate.

E4.2 SIP format

The SIP Packet Length is from 24 to 255 data slots, in the format specified in Table E1. Equipment that implements SIPs shall be required to transmit them at least once every 15 seconds.

Table E2 - SIP Format

Slot	Definition	Refer to Clause
1	SIP Byte Count/SIP Checksum pointer (valid values in the range of 24 - 255)	
2	Control Bit Field	E4.3
3	MSB of Optional 16 bit additive Checksum of previous packet	E4.4
4	8 bit additive or the LSB of the Optional 16 bit Checksum of previous packet	
5	SIP sequence number	E4.5
6	DMX512 universe number	E4.6
7	DMX512 processing level	E4.7
8	Version of Software sending this SIP	E4.8
9	Standard Packet Len MSB	
10	Standard Packet Len LSB	E4.9
11	Number of Packets transmitted by originating device since last SIP MSB	
12	Number of Packets transmitted by originating device since last SIP LSB	
13	Originating Device's Manufacturer ID MSB	E4.10
14	Originating Device's Manufacturer ID LSB	E4.11
15	Second Device's Manufacturer ID MSB	
16	Second Device's Manufacturer ID LSB	E4.12
17	3 rd Predecessor to Last Device's Manufacturer ID MSB	
18	3 rd Predecessor to Last Device's Manufacturer ID LSB	
19	4 th Predecessor's Device Manufacturer ID MSB	
20	4 th Predecessor's Device Manufacturer ID LSB	
21	5 th Predecessor's Device Manufacturer ID MSB	
22	5 th Predecessor's Device Manufacturer ID LSB	
23	<i>reserved for future use</i>	
nn	Checksum of the SIP (max 255)	E4.13

E4.3 Control bit field

d7	d6	d5	d4	d3	d2	d1	d0
1 = MSB checksum exists	1=LSB checksum exists	reserved transmit as 0	reserved transmit as 0	reserved transmit as 0	reserved transmit as 0	reserved transmit as 0	control bit set=1

Processing of the Control Bit is optional. If implemented, when the Control Bit is set (=1), the subsequent NULL START Code packet shall be held pending the reception of the next SIP for validation of the checksum. If a second NULL START Code packet is received without a preceding SIP, the receiver shall return to an immediate use mode and flag this as an error condition.

E4.4 Checksums

8 bit additive checksum (or optional 16 bit additive checksum) of all slots in the previous packet. The checksum includes the START Code. Bits must be set in the control bit field to indicate which type of checksum is being sent.

E4.5 SIP Sequence number

A free running 8 bit counter identifying the SIP and incremented by a SIP generator by 01h on every subsequent SIP. This field may be checked to ensure that SIPs are not being dropped randomly.

E4.6 Originating universe

This slot indicates the (originating) DMX512 universe currently transmitted on this link. 00h is not used. Valid values 01h - FFh (1 decimal - 255 decimal).

E4.7 DMX512 processing level

This slot indicates the level of post console processing. Originating devices shall always transmit a value of 00h in this field. Processing devices such as merge units or any that regenerate or provide a media conversion (e.g., Ethernet to DMX512) facility shall increment the value of this field by 01h. The content of this field indicates a level of process “hops” that data on the link has been subjected to relative to the originating transmitting device.

E4.8 Software version

00h not implemented

01h - FFh firmware version of last device

NOTE: This slot for use by the manufacturer and may not correlate with any formally published release identifier.

E4.9 Packet lengths

This declares the standard length of packets for START Code 00, normally transmitted on this link.

Valid values are

0000h	packet length not declared
0001h - 0200h	designates value of fixed packet length
0201h - 7FFFh	are not used
8000h	Dynamic Packet, length not declared
8001h - 8200h	length of last dynamic packet
8201h - FFFFh	are not used

E4.10 Number of packets

A 16 bit count of the number of packets transmitted by the originating device since last SIP was transmitted. This count should not increment past FFFFh.

E4.11 Manufacturer ID

Manufacturer ID will be the same 16 bit assignment as used for the Manufacturer's ID field used with Alternate START Code 91h (see Annex E - clause E1).

an ID == 0000h indicates that Manufacturer is not declared.

an ID == FFFFh indicates that Manufacturer has applied for, but not been granted, an ID and that this transmission originates from a product under development.

E4.12 Packet history

Specialist DMX512 Processing devices and media converters shall be required to insert their own Manufacturer's ID into the next available SIP slot. An originating device shall always send its Manufacturer's ID in SIP slots 13 and 14, with 0000h in slots 15, 16; 17, 18; 19, 20 and 21, 22. The next downstream processing device shall insert its own Manufacturer's ID into slots 15, 16. The next+1 downstream processing device shall insert its own Manufacturer's ID into slots 17, 18 and so on.

NOTE: This scheme allows for a packet processing history to be traced back through a complex installation of products.

E4.13 SIP Checksum

8 bit additive checksum of the SIP START Code (CFh) and first 23 slots of SIP data.

-end of Annex E-

This Page Almost Blank

Annex F (Normative) - Alternate START Code, Manufacturer ID, and Enhanced Functionality Registration

F1 Alternate START Code Registration Policy: 1 - 255 decimal (01 - FF hexadecimal).

Slot 0 of a DMX512 packet is the START Code. The value of this slot identifies intended use of data in the rest of the packet. The Standard provides for a non NULL or "Alternate" START Code. Where it is required to send proprietary information over a DMX512 data link, a packet starting with a registered Alternate START Code shall be used.

F2 Authorized use

The E1 Accredited Standards Committee or any organization that it authorizes may use an Alternate START Code to provide further extensions to the DMX512 Standard.

F3 Reserved Alternate START Codes

Several Alternate START Codes are reserved for special purposes or for future development of the Standard. See Annex E.

F4 Requests for Registration of New START Codes

Any manufacturer or organizations involved in the use of DMX512 may request that a START Code or Manufacturer ID be registered for their use. Although not encouraged, an Alternate START Code and Manufacturer ID may be registered for proprietary use. Requests shall be forwarded to the Secretariat for the ANSI E1 Accredited Standards Committee – ESTA. ESTA will attempt to honor such reasonable requests as described below.

F4.1 Number of Alternate START Codes per entity

No more than one Alternate START Code may be registered to any one manufacturer/organization. Manufacturers and Organizations with Alternate START Codes registered prior to the publication of this Standard may request one additional Alternate START Code.

F4.2 Selection of the Alternate START Code value and Manufacturer ID

The assignment of any particular numeric START Code or Manufacturer ID value to any particular entity is solely at the discretion of Accredited Standards Committee E1. Assignment depends on the availability of unused and unreserved START Codes and Manufacturer IDs.

F5 Requirements for registration of an EF protocol

No EF protocol intended for use between multiple Manufacturer's will be registered as an EF protocol unless its basic structure is available to anyone by request and can be used freely by any manufacturer. The entire message structure for the protocol does not have to be made public, only the portions that are for public use. It is expected that some protocols may still have portions of the message structure that are reserved for proprietary use (i.e., vendor specific messages). Registration is at the discretion of Accredited Standards Committee E1.

F6 Documentation Register

F.6.1 Documentation for use of Alternate START Codes

The manufacturer/organization requesting registration of an Alternate START Code shall provide a 2-line description of the purpose of the Alternate START Code. They shall list the minimum and maximum number of slots, including the START Code, in any proposed packet. Any provided description (subject to editing) shall be included in the Register. This is not required for functions associated with a manufacturer specific ID under Alternate START Code 91. It is recommended, but not required, for Alternate START Codes assigned prior to the adoption of this version of the Standard.

F6.2 Maintenance and Publication

The ANSI E1 Accredited Standards Committee through its secretariat (ESTA) shall maintain a Register of Alternate START Codes and Manufacturer IDs. ESTA will publish the Registry as needed.

F6.3 Supplemental documentation

If the manufacturer/organization wishes detailed documentation to be in the Public Domain, a note will be added to the Registry, but they will be responsible for such publication.

F7 Ownership

The DMX512 Standards are copyrighted. By registering a START Code or Manufacturer ID, no ownership rights are conferred to any third party. Alternate START Codes are registered to particular entities solely to allow for orderly management of the Standard. The registrant does not own the Alternate START Code or Manufacturer ID.

-end of Annex F-

Inside Back Cover

ESTA Technical Standards Manager:
Karl Ruling

ESTA Control Protocols Working Group - Co-Chairs:
Steve Carlson, High Speed Design
Steve Terry, Production Arts - Production Resource Group

ESTA Control Protocols Working Group - DMX512 Task Group Members:
Chair/Editor: Mitch Hefter, Rosco / ET; USITT (USA)

Canada
Dave Higgins, Gray Interfaces

USA
Tim Bachman, Barbizon Light
Steve Carlson, High Speed Design
Milton Davis, Strand Lighting
Doug Fleenor, Doug Fleenor Design
Bob Goddard, Goddard Design
Ted Paget, Jones & Phillips
Steve Terry, Production Arts - Production Resource Group

Germany
Eckart Steffens, Soundlight; VPLT

UK
Tim Cox, PLASA
Tony Douglas-Beveridge, PLASA
Wayne Howell, Artistic Licence
Paul Mardon, Pulsar Light of Cambridge
Steve Unwin, Pulsar Light of Cambridge
Peter Willis, Andera Ltd.

Back Cover

Entertainment Services and
Technology Association



American National Standard
E1.11 - 2004
Entertainment Technology
USITT DMX512-A
Asynchronous Serial Digital Data
Transmission Standard for Controlling
Lighting Equipment and Accessories

CP/1998-1031r8.0

Entertainment Services and Technology Association



American National Standard E1.11 - 2004 Entertainment Technology USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories CP/1998-1031r8.0

This edition of ANSI E1.11 was approved by American National Standards Institute on November 8, 2004.

©2004 ASC E1, Safety and Compatibility of Entertainment Technical Equipment and Practices, and its secretariat the Entertainment Services and Technology Association. All rights reserved. No part of this publication may be reproduced in any material form (including photocopying or storing by electronic means) without the written permission of the copyright holder. Any parties wishing to translate and publish this document in another language must receive permission from the copyright holder.

Notice and Disclaimer

ESTA and ANSI Accredited Standards Committee E1 (for which ESTA serves as the secretariat) do 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, or an American National Standard developed under Accredited Standards Committee E1 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 or Accredited Standards Committee E1.

ESTA and ANSI Accredited Standards Committee E1 (ASC E1) neither guaranty nor warrant 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 and ASC E1 do 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 judgement 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**

875 Sixth Avenue, Suite 1005
New York, NY 10001 USA
Phone: +1-212-244-1505
Fax: +1-212-244-1502
Email: standards@esta.org

For Electronic Copies of this Document Contact:

ANSI

<http://webstore.ansi.org>

For Additional Copies of this
Document Contact:

ESTA Publications

c/o USITT
6443 Ridings Road, Suite 134
Syracuse, NY 13206 USA
Phone: +1-315-463-6463
Fax: +1-315-463-6525

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, PLASA, 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 as Accredited Standards Committee E1, Safety and Compatibility of Entertainment Technical Equipment and Practices.

The Technical Standards Committee (TSC) was established by ESTA's Board of Directors to oversee and coordinate the Technical Standards Program. Made up of individuals experienced in standards-making work from throughout our industry, the Committee approves all projects undertaken and assigns them to the appropriate working group. The Technical Standards Committee employs a Technical Standards Manager to coordinate the work of the Committee and its working groups as well as maintain a "Standards Watch" on behalf of members. Working groups include: Camera Cranes, Control Protocols, Electrical Power, Floors, Fog and Smoke, Photometrics, and Rigging.

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 two 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 manufacturers, consultants, dealers, and end-users. ESTA is committed to developing consensus-based standards and recommended practices in an open setting. Future Control Protocols Working Group projects will include updating this publication as changes in technology and experience warrant, as well as developing new standards and recommended practices for the benefit of the entertainment industry.

The United States Institute for Theatre Technology, Inc. (USITT) is the Association of Design, Production, and Technology Professionals in the Performing Arts and Entertainment Industry. Founded in 1960, the mission of the Institute is to advance the professions of design and technology in the performing arts by disseminating information, actively promoting the advancement of knowledge and skills and facilitating national and international communication among its members. USITT is the United States Center of OISTAT, the International Organization of Scenographers, Theatre Architects and Technicians.

USITT
6443 Ridings Rd.
Syracuse, NY 13206-1111
(800) 93USITT (315) 463-6463 (315) 463-6525 FAX
<http://www.usitt.org>

Contents

Foreword	vi
1 General	1
1.1 Scope	1
1.2 Overview and Architecture	1
1.3 Appropriate uses of this Standard	1
1.4 Classes of data appropriate for transmission over links designed to this Standard	2
1.5 Classes of data not appropriate for transmission over links designed to this Standard	2
1.6 Compliance	2
2 Normative references	2
3 Definitions	3
4 Electrical Specifications and Physical Layer	6
4.1 General	6
4.2 Electrical isolation	6
4.3 Topology	6
4.4 DMX512 ports	6
4.5 Data link common and grounding topologies	7
4.6 Preferred method of earth grounding data link common	7
4.7 Primary data link	7
4.8 Secondary data link	7
4.9 Data Link termination procedures	7
4.10 Unpowered devices	8
5 Nominal Operating Characteristics	8
5.1 General	8
5.2 Chassis in power isolated equipment	8
5.3 Earth grounding of data link common for transmitters	8
5.4 Ground referenced transmitters	8

5.5 Disallowed transmitter topology	10
5.6 Earth grounding of data link common for receivers	10
5.7 Isolated Receiver characteristics	10
5.8 Disallowed receiver topology	12
5.9 DMX512 Processing devices	12
5.10 Loading designation	12
6 Protection	13
6.1 Minimum protection against interconnection damage	13
6.2 Minimum Electro Static Discharge (ESD) protection	13
7 Connection Methods	13
7.1 Equipment fitted with user accessible pluggable data link connections	13
7.2 Equipment intended for fixed installation with internal connections to the data link	14
7.3 IEC 60603-7 8-position modular connectors	14
8 Data Protocol	15
8.1 Format	15
8.2 Slot format	15
8.3 Break	16
8.4 Mark After Break	16
8.5 START code	16
8.6 Maximum number of data slots	18
8.7 Minimum number of data slots	18
8.8 Defined line state between slots	18
8.9 Defined line state between data packets (Mark Before Break)	18
8.10 Break-to-Break spacing	18
8.11 Timing Diagram - data+	19
9 Receiver Performance	20
9.1 Rejection of improperly framed slots	20

9.2 Loss of data tolerance / Resumption of acceptance of data	20
9.3 Receiver performance at maximum refresh rate	21
9.4 Packet processing latency	21
10 Marking and disclosures	21
10.1 Identification	21
10.2 DMX512 port marking	21
10.3 Data line termination marking	22
10.4 Ground / Isolation marking	22
10.5 Required disclosures and markings	22
11 Protocol Implementation Conformance Statement (PICS) - Informative	23
11.1 Introduction	23
11.2 Implementation identification	23
11.3 PICS tables for USITT DMX512-A	25
Annex A - Non Preferred (Alternate) topologies (Normative)	32
A1 Isolated transmitters	32
A2 Non-isolated receivers	33
A3 Grounded Receivers	34
A4 Earth grounding of data link common for floating devices	35
Annex B (Normative) - Enhanced DMX512	37
B1 General	37
B2 Summary of Enhanced Function Topologies	37
B3 Identification of data protocols for Enhanced DMX512	38
B3.1 EF1 Half Duplex DMX512 - Bidirectional use of the primary data link	38
B3.2 EF2 - Full duplex DMX512	38
B3.3 EF3 Half duplex on the second data link	39
B3.4 EF4 Protocols that use the primary and the secondary data links in ways not covered above	40
B3.5 Additional electrical requirements	40
Annex C (Normative) - Higher Protection Levels – “DMX512-A Protected”	41
Annex D (Normative) - Reserved Alternate START Codes	42
D1 Reserved Alternate START Codes	42
D2 ASC text packet	42
D3 ASC test packet	43
D4 System Information Packet (SIP) Alternate START Code	43
D4.1 Application	43

D4.2 SIP format	43
D4.3 SIP checksum pointer	43
D4.4 Control bit field	44
D4.5 Checksums	45
D4.6 SIP Sequence number	45
D4.7 Originating universe	45
D4.8 DMX512 processing level	45
D4.9 Software version	45
D4.10 Packet lengths	45
D4.11 Number of packets	45
D4.12 Manufacturer ID	46
D4.13 Packet history	46
D4.14 SIP Checksum	46
Annex E (Normative) - Alternate START Code, Manufacturer ID, and Enhanced Functionality	
Registration	47
E1 Alternate START Code Registration Policy: 1 - 255 decimal (01 - FF hexadecimal)	47
E2 Authorized use	47
E3 Reserved Alternate START Codes	47
E4 Requests for Registration of New START Codes	47
E4.1 Number of Alternate START Codes per entity	47
E4.2 Selection of the Alternate START Code value and Manufacturer ID	47
E5 Requirements for registration of an EF protocol	47
E6 Documentation Register	48
E6.1 Documentation for use of Alternate START Codes	48
E6.2 Maintenance and publication	48
E6.3 Supplemental documentation	48
E7 Ownership	48
Annex F (Informative) - Protocol Implementation Conformance Statement (PICS) for Annexes A through E	
F1 Introduction	49
F2 Implementation identification	49
F2.1 Identification	49
F2.2 Protocol summary	49
F3 (PICS) tables for Annexes A through E	50
F3.1 Annex A – Non preferred (alternate) topologies	50

F3.2 Annex B – Enhanced DMX512	52
F3.3 Annex C – Higher Protection Levels (DMX512-A Protected)	56
F3.4 Annex D – Reserved Alternate START Codes	56
F3.5 Annex E – Alternate START Code, Manufacturer ID, and Enhanced Functionality Registration	59

Tables:

1 - Ground Referenced Transmitter Characteristics	9
2 - Isolated Receiver Characteristics	11
3 - Signal Designations Summary	13
4 - Connection Schedule for DMX512 Equipment Using IEC 60603-7 8-Position Modular Connectors	15
5 - Data Slot Format	15
6 - Timing Diagram Values - Output of Transmitting UART	20
7 - Timing Diagram Values for Receivers	20
8 - Signal Designations Abbreviations Allowed for Marking	21
9 - Ground / Isolation Marking	22
A1 - Isolated Transmitter Characteristics	33
A2 - Non-Isolated Receiver Characteristics	34
B1 - Enhanced Function Topologies	37
C1 - Transmitter Protection	41
C2 - Receiver Protection	41
D1 - Reserved Alternate START Codes	42
D2 - SIP Format	44

Figures:

1 - Ground Referenced Transmitter	9
2 - Transmitter Topology NOT Allowed	10
3 - Isolated Receiver	11
4 - Receiver Topology NOT Allowed	12
5 - Timing Diagram	19
A1 - Isolated Transmitter	32
A2 - Non-Isolated Receiver	33
A3 - Grounded Receiver	35
A4 - DMX512 Device, Floating	36

Foreword

(This foreword contains no requirements and is not part of E1.11)

This Standard describes a method of digital data transmission between controllers and controlled lighting equipment and accessories, including dimmers and related equipment. This Standard is intended to provide for interoperability at both communication and mechanical levels with controllers made by different manufacturers.

There are five normative annexes in this Standard. These address extensions of the base standard and are considered part of the Standard, which means that when an extension described in an Annex is implemented, compliance with the annex is mandatory. However, a product compliant with the Standard can be manufactured without implementing these annexes.

The original version of the DMX512 Standard was developed in 1986 by the Engineering Commission of the United States Institute for Theatre Technology, Inc. (USITT). Minor revisions were made in 1990. DMX512 has gained international acceptance throughout the entertainment industry, even though USITT is not formally accredited as a standards making body. The earlier versions of this Standard covered only data used by dimmers. In practice this Standard has been used by a wide variety of devices; this version recognizes this fact.

In 1998, it became evident that additional updates to the Standard were necessary and formal recognition through an internationally recognized standards organization was required. USITT issued a Call for Comments in order to solicit recommendations for changes to the Standard. At the same time, USITT transferred maintenance of DMX512 to ANSI Accredited Standards Committee *E1, Safety and Compatibility of Entertainment Technical Equipment and Practices* (more commonly known as the ESTA Technical Standards Program, or TSP).

A Task Group established under the TSP's Control Protocols Working Group acted on the proposals received in response to the Call for Comments. The primary goal was to make editorial updates to DMX512 appropriate for current times, including the addition of technical features while maintaining a balance with backward compatibility. Many proposals, while technically innovative, could not be accepted because their implementation would not have been backward compatible and would have immediately rendered obsolete most of the installed base of equipment.

This document is a result of the actions taken on those proposals and subsequent development under the *Policies and Procedures* of the ESTA Technical Standards Program. Despite being an American National Standard, development has had strong international participation and support.

1 General

1.1 Scope

This Standard describes a method of digital data transmission between controllers and controlled equipment as described in Clause 1.4 and accessories, including dimmers. It covers electrical characteristics, data format, data protocol, and connector types.

This Standard is intended as a guide for:

1. Equipment manufacturers and system specifiers who wish to integrate systems of lighting equipment and accessories, including dimmers, with controllers made by different manufacturers.
2. Equipment manufacturers seeking to implement a standard digital transmission protocol in their lighting control and accessory products.
3. System specifiers and designers to gain detailed information about allowed connectors and allowed system topologies.

This standard is not intended to replace existing protocols other than USITT DMX512 and DMX512/1990. Cable requirements and premises wiring are not within the scope of this standard.

Equipment compliant with this standard will be marked DMX512-A or USITT DMX512-A in order to distinguish it from the previous (informally recognized) versions. Unless otherwise noted, references to DMX512 in this document refer to DMX512-A.

1.2 Overview and Architecture

This standard uses a simple asynchronous eight-bit serial protocol consisting of an untyped byte stream produced by standard UARTs. The physical media, not addressed in this document, is normally, but not exclusively, a two-pair cable, with each pair serving as a data link. The media is driven using ANSI/TIA/EIA-485-A-1998 (hereafter referred to as EIA-485-A in this document) balanced data transmission techniques. Physical connection at devices is via 5-pin XLR connectors or by “hard-wiring” to terminals. Restricted use of connectors other than 5-pin XLR is allowed if certain conditions apply (see clause 7).

Data on the primary data link is sent in packets of up to 513 slots. The first slot is a START Code, which defines the information in the subsequent slots in the packet. The interoperability of equipment complying with the Standard is largely due to the use of the NULL START Code by transmitting devices. Proper function is dependent upon the receiving device(s) extracting the pertinent data for processing from each transmitted packet.

Data on the secondary data link, when implemented, is used for a variety of purposes, all of which fall within the scope of EIA-485-A. Identification of the required circuit topology for any particular implementation is defined.

1.3 Appropriate uses of this Standard

Equipment designers and general users of this Standard will recognize that this Standard is intended to fill only a limited range of uses. Other standards will be more appropriate for different uses.

This is not intended to support a venue wide network that can carry data for lighting, sound, and scenery mechanization, for example, all on the same wire.

This Standard does not require mandatory error checking of NULL START Code packets. There is no assurance that all DMX512 packets will be delivered. It is common practice for merge units and protocol convertors to drop packets that they cannot process in a timely manner. The 1986 and 1990 versions of the USITT Standard specifically allow dimmers to ignore packets that they cannot process in a timely manner, and this concept survives in this version of the Standard with respect to NULL START Code packets.

1.4 Classes of data appropriate for transmission over links designed to this Standard

DMX512 is designed to carry repetitive control data from a single controller to one or more receivers. This protocol is intended to be used to control dimmers, other lighting devices and related non-hazardous effects equipment.

1.5 Classes of data not appropriate for transmission over links designed to this Standard

Since this Standard does not mandate error checking, DMX512 is not an appropriate control protocol for hazardous applications.

1.6 Compliance

Compliance with this Standard is strictly voluntary and the responsibility of the manufacturer. Markings and identification or other claims of compliance do not constitute certification or approval by the E1 Accredited Standards Committee. See clause 10 for Marking and Disclosure requirements. See clause 11 and Annex F for Protocol Implementation Conformance Statements (PICS).

2 Normative references

ANSI/TIA/EIA-568-B-2001 *Commercial Building Telecommunications Cabling Standard*

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*

IEC 60603-7

*Connectors for Frequencies Below 3 MHz for Use with Printed Wiring Boards-
Part 7: Detail Specification for Connectors, 8-Way, Including Fixed and Free
Connectors with Common Mating Features, with Assessed Quality*

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

USITT DMX512/1990

Digital Data Transmission Standard for Dimmers and Controllers

USITT

6443 Ridings Rd.
Syracuse, NY 13206-1111
+1-800-938-7488 +1-315-463-6463 Fax: +1-315-463-6525
<http://www.usitt.org>

3 Definitions

3.1 Asynchronous: signals that start at any time and are not locked or synchronized to the receiving device by a separate clock line.

3.2 Balanced Transmission Line: a data communications line where two wires are present, the signal and its opposite (complement), the actual signal being the difference between the voltages on the two wires. Balanced lines have excellent noise and interference rejection properties.

3.3 Break: a high (mark) to low (space) transition followed by a low of at least 88 microseconds followed by a low to high transition.

3.4 Circuit Common: the common reference (zero volt supply) of the EIA-485-A driver or receiver circuitry.

3.5 Common: see Data Link, Signal Common, and Circuit Common.

3.6 Common Mode Voltage: a voltage appearing equally on the data+ (plus) and data- (minus) lines relative to circuit Common. $V_{cm} = (V_a + V_b)/2$ where:

V_{cm} is the Common Mode Voltage

V_a is the voltage on DMX512 data+ with respect to circuit Common

V_b is the voltage on DMX512 data- with respect to circuit Common

3.7 Controller: a transmitting device that originates DMX512 data.

3.8 Data+: signal true.

3.9 Data-: signal complement.

3.10 Data Link: physical connection between transmitting and receiving devices.

3.11 Data Link Common: the connection to circuit Common at the point of interconnection (DMX512 Port) of the product.

3.12 DMX512 Port: a DMX512 signal connection point (connector or terminal strip).

3.13 DMX512 Processing Device: a piece of equipment that regenerates the timing of any DMX512 packet or has provision for other signal inputs from which the outgoing DMX512 packet is generated. In the absence of any DMX512 transmitting capability, the device has provision for other signal outputs that are controlled in some manner by the incoming DMX512 packet. Basic buffer products are not normally considered processing devices.

3.14 Earth Ground: the common, zero potential available from the mains electricity supply and usually connected to the metal chassis of equipment. Earth Ground is referred to as Earth in Europe and Ground in the USA.

3.15 Enhanced Functionality: use of the optional secondary data link of a DMX512 port and/or optional additional use of the primary data link of a DMX512 port.

3.16 Idle: the time between slots that the DMX512 line is high and not sending any information (also known as the 'Mark' condition).

3.17 In-Line Device: any component that receives and re-transmits DMX512.

3.18 Isolated: circuit topology in which the output is completely electrically disconnected from the input.

3.19 Legacy (as used in this Standard): transmitting and receiving devices complying with the original USITT DMX512 or DMX512/1990 in all aspects of those standards. (Exception: receiving devices that are not dimmers but comply with all other aspects of DMX512/1990 are considered to be Legacy Equipment.)

3.20 Line Driver: an electrical circuit providing differential voltage excursions on a data link, operating within a defined Common Mode voltage range and with a specified response to overload and overvoltage conditions. Sometimes simply referred to as a "driver."

3.21 Line Receiver: an electrical circuit allowing detection of differential voltage excursions on a data link, operating within a defined Common Mode voltage range and with a specified response to overload and overvoltage conditions.

3.22 Loop-Through Connection: a connector or terminal DMX512 port that connects the signals Data 1+, Data 1- and Data link common, and optionally Data 2+ and Data 2-, of one DMX512 port to another DMX512 port. Frequently abbreviated to Loop or Thru.

3.23 Manufacturer ID: a two-byte value assigned to a Manufacturer/Organization by the E1 Accredited Standards Committee for use with specific Alternate START Codes. This ID identifies the data contained in the packet as proprietary. This packet may be safely ignored by systems that do not implement the specific start code.

3.24 Mark: a line condition where Signal True is high with respect to Signal Complement. A Mark represents a binary 1.

3.25 MAB (MaB): Mark After Break – the period of time between the low to high transition that signifies the end of Break and the high to low transition which is the start bit of the START Code (slot 0).

3.26 MBB (MbB): Mark Before Break – the period of time between the end of the second stop bit of the last slot and the high to low transition that signifies the start of Break.

3.27 Merge Unit: a product comprising one or more receiving devices and one or more transmitting devices that generate a DMX512 packet derived from the manufacturers declared logical combination of the DMX512 input packets.

3.28 NULL START Code: a START Code with a value of zero (00h).

3.29 Packet (in DMX512-A): a Reset Sequence followed by all slots up to the Mark Before Break.

3.30 Receiver (Receiving Device): a piece of equipment that accepts a DMX512 signal.

3.31 Refresh Rate: the number of DMX512 packets with a NULL START Code sent per second.

3.32 Reset Sequence: a sequence of a Break, Mark After Break, and START Code.

3.33 Signal Common: the common reference conductor of the physical media (e.g., the cable shield).

3.34 Slot: a sequentially numbered framed byte in a DMX512 packet. A single Universe contains a maximum of 513 Slots, starting at slot 0. Slot 0 is the START Code. Slots 1 through 512 are data slots.

3.35 Slot Footprint: the number of data slots used by a product in its operation.

Note: A 24 way dimmer rack may have a footprint of 24, it may be more if some slots are used to provide additional control functions using NULL START Code packets. Automated luminaires usually require a Slot Footprint of greater than one.

3.36 Space: a line condition where Signal True is low with respect to Signal Complement. A Space represents a binary 0.

3.37 Start Bit: the additional bit attached to the beginning of a byte to indicate to the receiver that a new byte is being sent. The start bit is always low, i.e., Space.

3.38 START Code: the first slot sent after Break, indicating the type of information to follow.

3.39 Stop Bit: the additional bit(s) attached to a byte to indicate the end of the byte – DMX512 has 2 stop bits. The stop bit is always high, i.e., Mark.

3.40 Terminator: a device or circuit topology that is designed to minimize unwanted signal reflections on a data link.

3.41 Transmitting Device: a piece of equipment that produces a DMX512 signal.

3.42 UART: Universal Asynchronous Receiver/Transmitter. A device that generates and/or decodes serially transmitted data.

3.43 Universe: a DMX512 data link originating from a single DMX512 source. Control of up to 512 DMX512 data slots comprises a single universe.

4 Electrical Specifications and Physical Layer

4.1 General

The electrical specifications of this Standard are those of EIA-485-A, except where specifically stated in this document. Where a conflict between EIA-485-A and this document exists, this document is controlling as far as this Standard is concerned.

The physical layer of a DMX512 data link is constrained by earth grounding practices, termination-methods, signal levels, EMC, and accidental damage by connection to other devices.

In addition to complying with the requirements of EIA-485-A, clauses 5 and 6 specify additional requirements not addressed by EIA-485-A.

4.2 Electrical isolation

EIA-485-A makes no general provisions for electrical isolation, however, this Standard does. Suitable optical isolation, transformer isolation, or other means may be employed to prevent the undesirable propagation of voltages that exceed the Common Mode limits of EIA-485-A (see clauses 5 and 6). The inclusion of such isolation does not, however, alter the requirement that a transmitter or receiver conforms to EIA-485-A.

4.3 Topology

A data link shall consist of a single active differential line driver, a terminated transmission line and one or more differential line receivers meeting the requirements of EIA-485-A and all additional requirements of this Standard. A DMX512 controlled device capable of returning status information is referred to as a responder in this standard. For the purposes of this standard, responders are considered to fall under the category of receiving devices. This subject is covered in Annex B.

4.4 DMX512 ports

A DMX512 Port is the signal connection point between the internal electronics of a device and the physical transmission line (cable), referred to as the equipment interconnect points by EIA-485. It may be made either by the prescribed connector as defined in clause 7 or by a terminal strip. A DMX512 port carries four signals and a common reference, designated data 1-, data 1+, data 2-, data 2+, and data link common. (In some situations, data 2- and data 2+ are not connected / provided). Historically, these signals have been referred to as Pins which correlates to the physical pinout used on the XLR style connectors as defined in clause 7 of this Standard. There are other situations where different physical connections may be encountered, such as a terminal strip.

4.5 Data link common and grounding topologies

Various portions of clause 5 and Annex A deal with shield-to-earth ground topologies. In all cases there is a low impedance connection between data link common pin or contact of the DMX512 port and signal common of the EIA-485-A driver or receiver circuitry.

4.6 Preferred method of earth grounding data link common

DMX512 systems should make use of earth ground referenced transmitting devices and isolated receiving devices. This approach provides for a single point solid ground/chassis connection at the source, and allows for variations in building ground potentials between transmitting and receiving devices. This is to ensure that interoperability of equipment is achieved in situations that might otherwise exceed the Common Mode limitations of EIA-485-A. See EIA-485-A clause 4.3.1. Other approaches are covered in Annex A.

4.7 Primary data link

Data 1- and Data 1+ of a DMX512 Port form the primary data link. Format of the data is covered in clause 8. Limited use of multiple data link drivers for half-duplex, bi-directional data transmission on the primary data link is permitted in accordance with Enhanced Functionality as described in Annex B.

4.8 Secondary data link

Implementation of the secondary data link is optional.

4.8.1 Secondary data link - active use

Data 2- and Data 2+ of a DMX512 Port provide a secondary EIA-485-A data link. Implementation of this data link is optional. Several different network topologies are associated with the implementation of the secondary data link. Use of the secondary data link is permitted only in accordance with Enhanced Functionality as described in Annex B.

4.8.2 Secondary data link - passive loop through ports

In order to extend Enhanced Functionality across a network, devices containing two DMX512 ports, one for receive and one for transmit, that do not actively process or buffer data, shall provide a direct passive link for all signals between the two ports. Devices containing three or more DMX512 ports may provide a passive link between only two of the ports. These two ports shall be declared as required in Clause 10 and should be marked on the product. In order to enhance interoperability of DMX512 products, equipment designers are encouraged to provide passive loop through on the secondary data link pins or contacts whenever possible, even if not required per this clause.

4.9 Data link termination procedures

DMX512 data links shall be terminated to eliminate ringing and signal reflection, which can cause mis-operation of an otherwise properly designed system. To comply with this Standard, all equipment connected to a DMX512 data link shall operate in accordance with the stated manufacturer's specification when the data link is terminated. The terminator shall be a 120 ohm +5%/-10% impedance placed between Data+ and Data-. In the preferred topology where the transmitter is connected

to one end of the data link, the far end of the data link shall be terminated. In the case where the transmitter cannot be connected at one end of the data link, then both ends of the data link shall be terminated. Manufacturers of receiving devices may provide internal termination of the data link. Where such termination is provided, it shall comply with the electrical and marking requirements of this Standard. It is recommended that termination components be chosen to withstand continuous voltages of at least 30 VAC 50 Hz/42 VDC.

4.10 Unpowered devices

Unpowered connected DMX512 devices shall not degrade the performance of the DMX512 transmission system nor materially lower the impedance they present to the data link.

5 Nominal Operating Characteristics

5.1 General

Operation limits generally follow the detailed requirements of EIA-485-A. Where appropriate, separate limits are given for isolated products. All electrical characteristics shall be measured at the DMX512 ports of the product.

5.2 Chassis in power isolated equipment

In equipment where connection to protective grounding is not made concurrent with power connection, such as battery powered equipment or equipment powered by isolated low voltage transformers, chassis is deemed to include any exposed metal DMX512 connector parts which do not carry signals.

5.3 Earth grounding of data link common for transmitters

In recognition of the need for DMX512 compliant products to be capable of interconnection as part of large and potentially complex systems, this Standard defines two allowable topologies for the earth grounding of data link common and circuit common for transmitters, to be known as "Ground Referenced" and "Isolated". The preferred method is "Ground Referenced." "Isolated" is covered in Annex A.

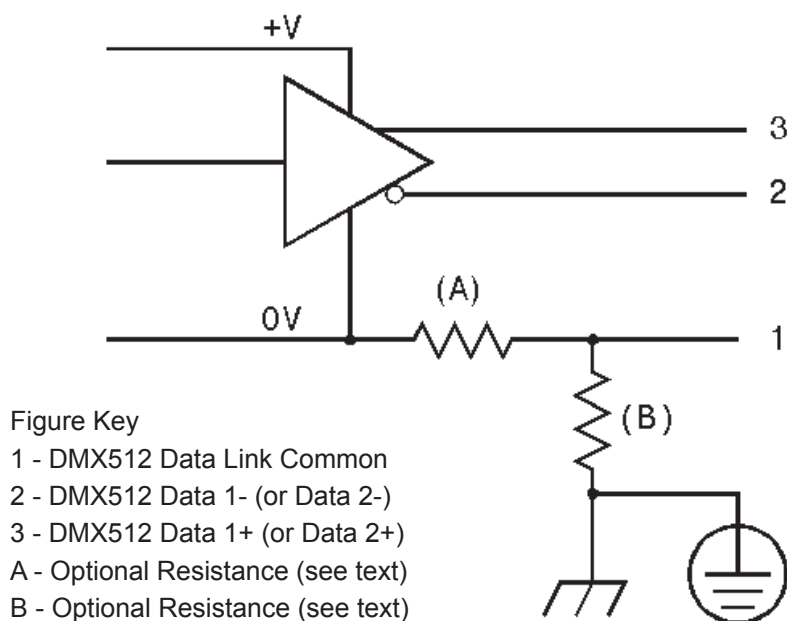
5.4 Ground referenced transmitters

Ground referenced transmitting device outputs shall meet the following conditions in table 1 during normal operation under open circuit condition.

Table 1 - Ground Referenced Transmitter Characteristics

Connection	Limit (measured at port pin or contact)	Comment
Data 1+ to Data Link Common or Data 1- to Data Link Common	$0 \leq v \leq +6 \text{ VDC}$	
Data 2+ to Data Link Common or Data 2- to Data Link Common	$0 \leq v \leq +6 \text{ VDC}$	Enhanced Function Devices Only
Data Link Common to Chassis	0V	
Data 1- to Chassis or Data 1+ to Chassis	$0 \leq v \leq +6 \text{ VDC}$	
Data 2- to Chassis or Data 2+ to Chassis	$0 \leq v \leq +6 \text{ VDC}$	Enhanced Function Devices Only
Data 1- to Data1+	+/- 6V (open circuit)	
Data 2- to Data 2+	+/- 6V (open circuit)	

Figure 1 illustrates a ground referenced transmitter port. It is characterized by the direct connection of the shield (Data Link Common) to chassis and protective earth. Therefore, devices employing ground referenced transmitters shall be provided with provision for connection to protective earth. Any resistance (A) between Data Link Common's pin or contact and zero volt supply (circuit common) of the transmitter circuit shall be less than 100 ohms. Any resistance (B) between Data Link Common's pin or contact and chassis shall be less than 20 ohms and is preferably zero ohms.

**Figure 1 - Ground Referenced Transmitter**

A DMX512 device may have any number of Ground Referenced transmitter ports. Ground Referenced transmitter ports may be used by all DMX512 devices including ones that provide any number of non-DMX512 input or output ports. Adherence to this topology allows a DMX512 transmitter connector to be marked as shown in table 9.

Because the transmitter in this topology is grounded, the existence of an isolation barrier between the transmitter and any other part of the device shall NOT qualify output for marking as ISOLATED.

5.5 Disallowed transmitter topology

This configuration is not permitted, although it may exist on some legacy products. While this topology is described as one possible topology in EIA-485-A, it is not appropriate when considering operation of DMX512 transmitting devices in systems encountering differential ground potentials.

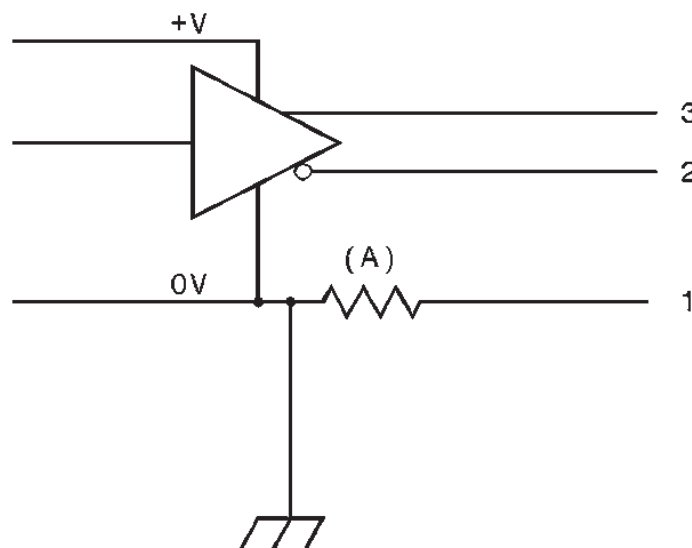


Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- A - $R > 0.2 \text{ ohm}$

Figure 2 - Transmitter Topology NOT Allowed

5.6 Earth grounding of data link common for receivers

This Standard defines several allowable topologies for earth grounding of data link common and circuit common for receiving devices. These are to be known as “non-isolated” and “isolated”. The preferred method is “isolated.” A specific concession is available to manufacturers of non-isolated receivers who, for reasons beyond the scope of this Standard, require a direct link between data link common and chassis. Non-isolated and grounded receivers are addressed in Annex A.

5.7 Isolated receiver characteristics

For isolated devices, a capacitor may be fitted between Data Link Common and chassis for the purpose of Radio Frequency bypass. Devices shall continue to operate correctly when exposed to any of the conditions in table 2.

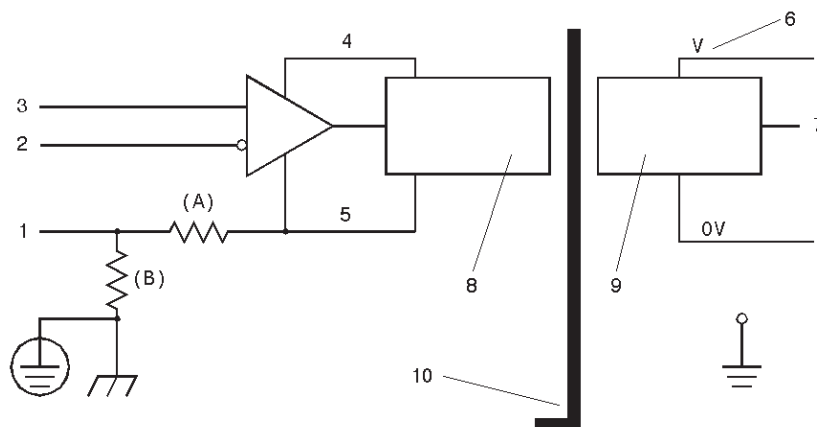
Table 2 - Isolated Receiver Characteristics

Connection	Limit (measured at port pin or contact)	Comment
Data 1+ to Data Link Common or Data 1- to Data Link Common	+12 / -7 VDC	Common Mode Range
Data 2+ to Data Link Common or Data 2- to Data Link Common	+12 / -7 VDC	Enhanced Function Devices Only
Data Link Common to Chassis	$\geq 22\text{M ohm @ } 42\text{ VDC}$	
Data 1- to Chassis or Data 1+ to Chassis	$\geq 22\text{M ohm @ } 42\text{ VDC}$	
Data 2- to Chassis or Data 2+ to Chassis	$\geq 22\text{M ohm @ } 42\text{ VDC}$	Enhanced Function Devices Only
Data 1- to Data1+	+/- 6V	
Data 2- to Data 2+	+/- 6V	Enhanced Function Devices Only
Any Port Pin or Contact to Chassis	30 VAC / 42 VDC	

Figure 3 illustrates an isolated receiver. Any signal pin or contact of the DMX512 isolated receiver shall present a resistance (B) greater than 22 Mohm at 42 VDC with respect to Chassis, with respect to Protective Ground (where fitted), with respect to any other signal inputs or outputs, and with respect to other ground referenced electronics. There may be a capacitance (not shown) between Data Link Common and chassis for Radio Frequency bypass. Any resistance (A) between Data Link Common's pin or contact and zero volt supply (circuit common) supply of the receiver circuit shall be less than 100 ohms.

Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- 4 - Isolated Supply
- 5 - Isolated 0V Supply
- 6 - V (+ or -)
- 7 - I / O
- 8 - Isolated Electronics
- 9 - Optional Non-Isolated Electronics
- 10 - Isolation Barrier
- A - Optional Resistance (see text)
- B - Optional Resistance (see text)

**Figure 3 - Isolated Receiver**

Adherence to this topology allows a DMX512 receiver port to be marked as ISOLATED. Isolated receiver ports may be used by all DMX512 devices including ones that provide any number of non-DMX512 input or output ports. A DMX512 device may have any number of isolated receiver ports.

5.8 Disallowed receiver topology

This configuration is not permitted, although it may exist on some legacy products. While this topology is described as one possible topology in EIA-485-A, it is not appropriate when considering operation of DMX512 receiving devices in systems encountering differential ground potentials.

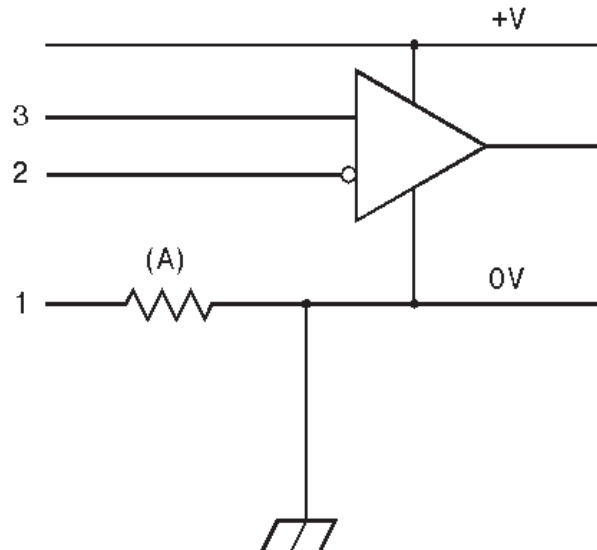


Figure Key

1 - DMX512 Data Link Common

2 - DMX512 Data 1- (or Data 2-)

A - $R \geq 0.2 \text{ ohm}$

Figure 4 - Receiver Topology NOT Allowed

5.9 DMX512 processing devices

It is permissible to design processing devices based on the Isolated Receiver / Ground Referenced Transmitter or Isolated Receiver / Isolated Transmitter models (see Annex A) already described.

5.10 Loading designation

As per EIA-485-A, the total load permitted on a DMX512 data link is 32 unit loads. Transmitters designed for this Standard shall be capable of driving 32 unit loads on a DMX512 data link. Each receiving port on a DMX512 device shall have a unit load of 1 or less as per EIA-485.

A receiver biased to any voltage from -7 to +12 volts shall not present a capacitive load to the line of more than 125 pF per unit load.

Declaring or marking of the unit load is not required by this standard. If a manufacturer chooses to declare or mark their products with a unit load value, the declared or marked value shall be the greater of either the DC unit load determined by EIA-485-A clause 4.1 or the unit load as determined by the capacitive loading. In either case, if the unit load is declared the capacitive load values shall also be declared.

6 Protection

6.1 Minimum protection against interconnection damage

Clause 4.2 of EIA-485-A recognizes that certain other extraneous conditions may overstress the system and that these conditions should be specified in the referencing standard. Extensive use of temporary and portable equipment in Entertainment Lighting Industry results in frequent connection and disconnection of equipment and gives rise to the possibility of equipment misconnection.

Equipment may be protected against damage resulting from accidental connection to voltages in excess of the minimum defined in EIA-485-A clauses 4.2.5 and 4.2.6, and this is recommended. See Annex C for requirements allowing DMX512 ports to be declared "DMX512-A Protected". This does not negate the need to comply with EIA-485-A clause 4.2.6 - Transient overvoltage tolerance.

6.2 Minimum electro static discharge (ESD) protection

Manufacturers shall ensure that any port pin or contact can withstand a minimum of 4kV ESD for contact discharge and 8kV ESD for air discharge in accordance with IEC 61000-4-2 or other local regulations that may require higher levels of protection. The acceptance criteria for this requirement allows for temporary loss of function, provided the function is self recoverable or can be restored by the operation of controls. Meeting this requirement does not alter the fact that a manufacturer may have to meet other more stringent ESD requirements to conform to local EMC regulations.

7 Connection methods

7.1 Equipment fitted with user accessible pluggable data link connections

This category includes all portable products.

Female connectors shall be used on controllers and other transmitting devices (primary data link) and male connectors shall be used on receiving devices. Female and male connectors consistent with this convention shall also be used where loop-through connections are provided.

7.1.1 Required connector

Equipment in this category shall use 5-pin XLR connectors with the physical pinout of the 5-pin XLR in accordance with table 3.

Table 3 - Signal Designations Summary

Use	5-Pin XLR Pin #	DMX512 Function
Common Reference	1	Data Link Common
Primary Data Link	2	Data 1-
	3	Data 1+
Secondary Data Link (Optional - see clause 4.8)	4	Data 2-
	5	Data 2+

7.1.2 Concession for use of an alternate connector (NCC DMX512-A)

A concession to use an alternate connector is available only when it is physically impossible to mount a 5-pin XLR connector on the product. In such cases all the following additional requirements shall be met :

- 1) The alternate connector shall not be any type of XLR connector.
- 2) The alternate connector shall not be any type of IEC 60603-7 8-position modular connector except as allowed in clause 7.3.
- 3) Provided that all other requirements of this Standard are met, in addition to the declaration in the equipment manual, the alternate connector shall be marked as NCCDMX512-A (Not Connector Compatible). If such a marking is not physically possible at the connector, an appropriate marking shall be made elsewhere on the equipment. The pin numbering on the alternate connector should match numbering for the standard 5-Pin XLR connector.
- 4) The manufacturer shall make available an adapter with the appropriate connections to a standard 5-pin XLR connector for all DMX512 ports included in the alternate connector.
- 5) The Enhanced Functionality, if applicable, and ground/isolation declarations shall continue to be declared for each DMX512 port.

7.2 Equipment intended for fixed installation with internal connections to the data link

Fixed installation products with internal connections to the data link may use the 5-pin XLR connector, but shall not use any other XLR connector. When use is made of the 5-pin XLR connector, the requirements of 7.1 and 7.1.1 shall apply. When a non-XLR connector is used, this Standard makes no other restriction or stipulation on connector choice. The contact (pin) numbering on the alternate connector should match numbering for the standard 5-Pin XLR connector.

Products in this category with Enhanced Functionality shall be marked in accordance with the provisions of clause 10 and Annex B.

7.3 IEC 60603-7 8-position modular connectors

The use of IEC 60603-7 8-position modular connectors (commonly referred to as RJ45 type connectors – plugs/jacks) and associated punchdown terminal blocks shall be limited to connections that are part of a fixed installation and not normally accessible except to qualified, authorized users, nor intended for regular connection and disconnection.

External (user accessible) IEC 60603-7 8-position modular connectors are permitted only on patch and data distribution products and only when permanently installed in controlled access areas.

Note: Examples of not normally accessible or controlled access areas include a locked electrical room or control booth, provided those who need access have a key (or lock combination) available.

Table 4 - Connection Schedule for DMX512 Equipment Using IEC 60603-7 8-Position Modular Connectors

Pin (Wire) #	Wire Color	DMX512 Function
1	white / orange	data 1+
2	orange	data 1-
3	white / green	data 2+ (optional)
6	green	data 2- (optional)
4	blue	Not assigned
5	white / blue	Not assigned
7	white / brown	Data link common (common reference) for data 1 (0 v)
8	brown	Data link common (common reference) for data 2 (0 v)
	drain	

Note 1: Pin numbering and color in accordance with ANSI/TIA/EIA-568 scheme T568B.

Note 2: Pin 8 should be wired as signal common even if pins 3 and 6 are NOT wired so that both conductors 7 and 8 are at equal potential.

Warning: Accidental connection to non-DMX512 equipment likely to be encountered (e.g., an Ethernet Hub at a patch bay) may result in damage to equipment. Pins 4 and 5 may carry voltages outside the EIA-485 range in telecom applications (e.g., telephone ringing). Pins 4 and 7 may carry voltages outside the EIA-485 range in other applications (e.g., some manufacturers whose distributed DMX512 buffering products require low voltage DC power may use these wires for this purpose). Because of these various uses, misplugging unlike systems could cause serious damage.

8 Data protocol

8.1 Format

DMX512 slots shall be transmitted sequentially in asynchronous serial format, beginning with slot 0 and ending with the last implemented slot, up to slot 512 (a maximum total of 513 slots). Prior to the first data slot being transmitted, a Reset Sequence shall be transmitted – a BREAK, followed by a MARK AFTER BREAK, and a START Code. Valid DMX512 data slot values under a NULL START Code shall be 0 to 255 decimal.

8.2 Slot format

The data transmission format for each data value transmitted are as shown in table 5. Note that no parity is transmitted.

Table 5 - Data Slot Format

Bit Position	Description
1	Start Bit, Low or SPACE
2 through 9	Slot Value Data Bits, Least Significant Bit to Most Significant Bit
10, 11	Stop Bits, High or MARK

8.3 Break

The BREAK indicates the start of a new packet.

The BREAK generated by a transmitter is defined as a mark-to-space transition followed by a low of at least the minimum duration shown in table 6 (Timing Diagram, Designation #1) followed by a low to high transition.

Table 7 (Timing Diagram, Designation 1) defines the minimum duration break a receiver is required to recognize as the start of a new packet.

8.4 Mark after break

The MARK separating the BREAK and the START Code (Timing Diagram, Designation #2) is defined as the MARK AFTER BREAK. DMX512 transmitters produce a MARK AFTER BREAK that complies with the minimum and maximum values for designator 2 in table 6. Compliant receivers correctly respond to data streams with a MARK AFTER BREAK at least as long as the minimum shown in table 7.

Note: The 1986 version of this standard specified a 4 microsecond MARK AFTER BREAK. The 1990 version of the standard changed that value to 8 microseconds, but added an option for receivers capable of recognizing the 4 microsecond MARK AFTER BREAK to be identified as having that capability. Some transmitters may still be in use that generate the shorter 4 microsecond MARK AFTER BREAK, and they may not work with equipment built to this standard.

8.5 START code

The START Code is the first slot (slot 0) following a MARK AFTER BREAK. The START Code identifies the function of subsequent data in that packet.

8.5.1 NULL START code

A NULL START Code identifies subsequent data slots as a block of un-typed sequential 8-bit information. Packets identified by a NULL START Code are the default packets sent on DMX512 networks. Earlier versions of this standard assumed that only dimmer class data would be sent using NULL START Code packets. In practice NULL START Code packets have been used by a wide variety of devices; this version recognizes this fact.

Each NULL START Code packet contains no formal data or addressing structure. The device using data from the packet must know the position of that data within the packet.

There is no guarantee that all NULL START Code packets will be delivered to all devices. Data sent using NULL START Codes should be of a type where loss of packet does not greatly affect the operation of the device. Hence data sent should be of the current value of a parameter, not a command to execute a routine. Once a controller is configured for a particular application, all NULL START Code packets should have the same number of slots.

8.5.2 Dimmer class data

Dimmer level data should be sent in NULL START Code packets. Valid dimmer levels shall be 0 to 255 decimal (00 to FF hexadecimal) representing dimmer control input. Value 0 shall represent a dimmer output of OFF or minimum and 255 shall represent an output of FULL. A dimmer shall respond to increasing the DMX512 slot value for 0 to 255 by fading from its minimum level (off) to its maximum level (full). The exact relationship between DMX512 slot values and dimmer output is beyond the scope of this Standard.

Note that NULL START Code packets are the default packets sent over DMX512 networks and may contain data other than dimmer class data.

8.5.3 Other START codes

In order to provide for future expansion and flexibility, DMX512 makes provision for 255 additional non NULL START Codes (1 through 255 decimal, 01 through FF hexadecimal), henceforth referred to as Alternate START Codes. Where it is required to send proprietary information over a DMX512 data link, a packet starting with a registered Alternate START Code shall be used.

Several Alternate START Codes are reserved. See Annex D.

See Annex E for Alternate START Code Registration Policies.

8.5.3.1 Alternate START code refresh interval

A DMX512 transmitter interleaving NULL START Code packets with Alternate START Code packets shall send a NULL START Code packet at least once per second.

8.5.3.2 Timing differences for Alternate START code packets

To ensure that in-line processing devices do not lose essential Alternate START Code data, a reduction of the maximum Alternate START Code update rate is necessary. One of the following methods shall be used:

- 1) Alternate START Code packets may be transmitted slower than the minimum timings specified in clause 8.11 by increasing the break to break time to 10% more than the minimum required for the Alternate START Code packets number of slots.
- 2) Transmitters may interlace non essential NULL START Code packets with Alternate START Code packets.

8.5.3.3 Handling of Alternate START code packets by in-line devices

DMX512 processing devices or any device that receives and re-transmits DMX512 shall state in the manual for the product how they process Alternate START Code packets. The acceptable processing methods are:

- 1) Block all packets containing particular Alternate START Codes. The START Codes blocked shall be declared (and may be all Alternate START Codes).
- 2) Pass unmodified all packets containing particular Alternate START Codes. The START Codes passed shall be declared.
- 3) Process the information contained in packets containing particular Alternate START Codes. The algorithm shall be declared in enough detail to allow the user to decide if the device will satisfy their needs.

DMX512 in-line repeating transmitters shall not pass some packets with a particular Alternate START Code while blocking other packets containing the same Alternate START Code unless doing so as part of a stated processing algorithm.

8.5.4 START code processing

All receiving devices other than in-line processing devices shall process the START Code and differentiate between those packets with NULL START Codes and those with Alternate START Codes. Devices shall not ignore START Codes by assuming that all packets received are NULL START Code packets.

8.6 Maximum number of data slots

Each data link shall support up to 512 data slots. Multiple links shall be used where larger numbers of slots are required.

8.7 Minimum number of data slots

There shall be no minimum number of data slots on the data link. DMX512 data packets with fewer than 512 slots may be transmitted, subject to the minimum timing requirements of this standard – see clause 8.10 and figure 5.

8.8 Defined line state between slots

The time between any two slots of a data packet (Timing Diagram, Designation #9) may vary between minimum and maximum values shown for designator #9 of table 6. The line shall remain in a marking state during any such idle period.

8.9 Defined line state between data packets (Mark Before Break)

Every data packet transmitted on the data link, regardless of START Code or length, begins with a BREAK, MARK AFTER BREAK, and START Code sequence known as a Reset Sequence (Timing Diagram - Figure 4, Item 12). The time between the second stop bit of the last data slot of one data packet and the falling edge of the beginning of the BREAK for the next data packet (Timing Diagram, Designation #10) may vary between minimum and maximum values shown for designator #10 of table 6. The line shall remain in a marking state throughout any such period. Transmitters, therefore, shall not produce multiple BREAKs between data packets. Receivers, however, shall be capable of recovering from multiple BREAKs produced by data link line errors.

8.10 Break-to-Break spacing

Transmitters produce packets so that the period between the falling edge at the start of any one BREAK and the falling edge at the start of the next BREAK are not be less than the minimum value in table 6 designator #13. This period is also not more than the maximum value in table 6 designator #13.

Receivers operate correctly when receiving packets with break to break spacing of at least the minimum value in table 7 designator #13 up to the maximum value in table 7 designator #13.

8.11 Timing diagram - data+

Timings shall follow the requirements of the timing diagram (figure 5) and its associated tables 6 and 7.

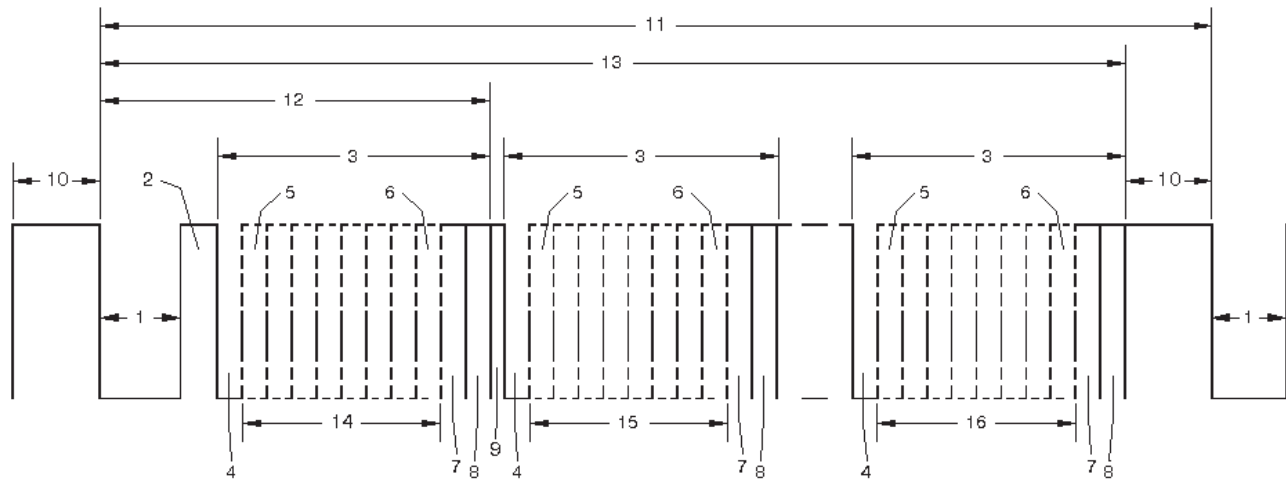


Figure Key

- 1 - "SPACE" for BREAK
- 2 - "MARK" After BREAK (MAB)
- 3 - Slot Time
- 4 - START Time
- 5 - LEAST SIGNIFICANT Data Bit
- 6 - MOST SIGNIFICANT Data Bit
- 7 - STOP Bit
- 8 - STOP Bit
- 9 - "MARK" Time Between Slots
- 10 - "MARK" Before BREAK (MBB)
- 11 - BREAK to BREAK Time
- 12 - RESET Sequence (BREAK, MAB, START Code)
- 13 - DMX512 Packet
- 14 - START CCODE (Slot 0 Data)
- 15 - SLOT 1 DATA
- 16 - SLOT nnn DATA (Maximum 512)

Figure 5 - Timing Diagram

Table 6 - Timing Diagram Values - Output of Transmitting UART

Designation	Description	Min	Typical	Max	Unit
-	Bit Rate	245	250	255	kbit / s
-	Bit Time	3.92	4	4.08	µs
-	Minimum Update Time for 513 Slots	-	22.7	-	ms
-	Maximum Refresh Rate for 513 Slots	-	44	-	updates / s
1	“SPACE” for BREAK	92	176	-	µs
2	“MARK” After BREAK (MB)	12	-	<1.00	µs s
9	“MARK” Time Between Slots	0	-	<1.00	s
10	“MARK” Before BREAK (MBB)	0	-	<1.00	s
11	BREAK to BREAK Time	1204	-	- 1.00	µs s
13	DMX512 Packet	1204	-	- 1.00	µs s

Table 7 - Timing Diagram Values for Receivers

Designation	Description	Min	Typical	Max	Unit
-	Bit Rate	245	250	255	kbit / s
-	Bit Time	3.92	4	4.08	µs
-	Minimum Update Time for 513 Slots	-	22.7	-	ms
-	Maximum Refresh Rate for 513 Slots	-	44	-	updates / s
1	“SPACE” for BREAK	88	176	-	µs
2	“MARK” After BREAK (MB)	8	-	<1.00	µs s
9	“MARK” Time Between Slots	0	-	<1.00	s
10	“MARK” Before BREAK (MBB)	0	-	<1.00	s
11	BREAK to BREAK Time	1196	-	- 1.25	µs s
13	DMX512 Packet	1196	-	- 1.25	µs s

9 Receiver performance

9.1 Rejection of improperly framed slots

A receiver shall check the first stop bit and should check the second stop bit of all received slots to determine if they have the correct value. If a missing stop bit is detected, the receiver shall discard the improperly framed slot data and all following slots in the packet.

9.2 Loss of data tolerance / Resumption of acceptance of data

A receiver not receiving a Reset Sequence (a sequence of a Break, Mark After Break, and START Code) within one maximum break to break time (as per table 7) of the previous Reset Sequence is considered to have lost data input.

Although this Standard does not specify loss of data handling procedures, manufacturers shall state what their Loss of Data handling procedures are.

Note: In the absence of overriding safety issues or an alternative source of control data, when encountering a loss of data condition, a receiving device should remain in an operating condition for at least 60 seconds, awaiting resumption of the DMX512 signal.

9.3 Receiver performance at maximum refresh rate

Any device incorporating a DMX512 receiver shall operate correctly when receiving continuous transmission of valid data packets containing any number of valid slot values.

9.4 Packet processing latency

Some products may provide their specified functionality without processing or being able to process every consecutive DMX512 packet. Such products will have an inherent latency to data changes between packets, which shall be declared by the manufacturer in accordance with the disclosure requirements of clause 10.

10 Marking and disclosures

In this standard, marking is defined as the application of a physical mark/label, etc. on the product, and declaration means declared in a manual and optionally marked on the product.

10.1 Identification

Only equipment conforming to this Standard may be marked and identified with “USITT DMX512-A” or “DMX512-A”.

10.2 DMX512 port marking

Where required by clauses by this standard, all DMX512 ports shall be marked as to the applicable Enhanced Functionality as defined in table B1 (Annex B). All information provided by marking shall also be provided in the equipment manual.

Manufacturers shall use the signal designations from table 3 in any pinout detail declaration or marking of pin, contact or terminal functions appearing on or within a product and associated with installation or connection. Where it is necessary to use abbreviations, only those detailed in table 8 shall be permitted.

Table 8 - Signal Designations Abbreviations Allowed for Marking

Function	Preferred Abbreviation	Non Preferred Abbreviation
Common Reference	COM	C
Primary Data Link - Data 1-	D1-	1-
Primary Data Link - Data 1+	D1+	1+
Secondary Data Link - Data 2-	D2-	2-
Secondary Data Link - Data 2+	D2+	2+




10.3 Data line termination marking

Products that have switchable (in/out) termination circuits shall provide indication of the termination state.

10.4 Ground / Isolation marking

All DMX512 ports shall indicate the relationship between Data Link Common and earth ground. The allowed grounding topologies are shown in table 9.

Table 9 - Ground / Isolation Marking

Function of DMX512 Port	Defining Clause / Figure	Comment	Approved Marking
Transmitter	Clause 5.4 / fig 1	Ground Referenced	no mark required, recommended mark 
Transmitter	Clause A1 / fig A1	Isolated (See Annex A)	ISO or Isolated
Transmitter	Clause A4 / fig A4	Floating (See Annex A)	FLT or Float or Floating
Receiver	Clause A2 / Fig A2	Non-Isolated, 100 ohm 2 Watt Resistor	NON-ISO or Non-Isolated
Receiver	Clause A3 / fig A3	Grounded - concession per clause A3	 
Receiver	Clause 5.7 / fig 3	Isolated	<i>no mark required</i> ; ISO or Isolated
Receiver	Clause A4 / fig A4	Floating	FLT or FLOAT or FLOATING

Note: Although no mark is required of Ground Referenced transmitters or Isolated receivers, it is highly recommended that such markings appear on the product.

10.5 Required disclosures and markings

10.5.1 Portable products and products fitted with external pluggable data link connectors

DMX512 ports on these products shall be marked in accordance with clause 10.2. DMX512 ports on these products shall provide Ground/Isolation marking in accordance with clause 10.4.

If use has been made of any non-XLR connector in conjunction with the supply of an adapter as permitted by clause 7.1.2, the non-XLR connector shall be declared in the equipment manual as NCC DMX512-A and also be marked (when feasible) in accordance with that clause.

10.5.2 Equipment intended for fixed installation with internal connections to the data link

Equipment intended for fixed installation with internal connections to the data link DMX512 ports on these products shall be marked in accordance with clause 10.2. DMX512 ports on these products shall provide Ground/Isolation marking in accordance with clause 10.4.

Clearly identified terminal contact (connector pinout) detail shall be marked on or within any product in accordance with clause 10.2.

10.5.3 Loss of data handling procedure

This Standard does not define loss of data handling procedures (clause 9) beyond requiring that manufacturers declare their own products' procedure(s). Such declarations shall be made in the equipment manual.

10.5.4 Packet processing latency

Manufacturers of in-line DMX512 processing, or distribution devices, shall declare any inherent latency in the processing of data packets, including a statement of the worst case delay between receiving a packet and that packet affecting the product's output. A statement of the worst case percentage number of zero start code packets that are discarded by the product shall also be included.

10.5.5 NULL START code functionality

Manufacturers of transmitting devices shall declare in the equipment manual the full range of slot values transmitted in conjunction with packets sent using the NULL START Code.

Manufacturers of receiving devices shall declare the response to packets received containing the NULL START Code, with particular reference to any functionality requiring limited or restricted slot data values.

10.5.6 Slot footprint

Manufacturers of receiving devices shall declare the slot footprint in the equipment manual.

11 Protocol Implementation Conformance Statement (PICS) - Informative

11.1 Introduction

The supplier of a protocol implementation that is claimed to conform to this Standard shall complete the following Protocol Implementation Conformance Statement (PICS).

11.2 Implementation identification

11.2.1 Identification

Manufacturer	
Contact information for inquiries about the PICS (Note 1)	
Implementation Name(s) and Version(s) (Notes 1,2)	
Other information necessary for full identification - e.g., name(s) and version(s) for equipment and/or operating systems; Systems Name(s) (Note 3)	
Manufacturer ID (if registered)	

Note 1: Required for all implementations.

Note 2: The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).

Note 3: May be completed as appropriate in meeting the requirements for the identification.

11.2.2 Protocol summary

Identification of the Protocol	ANSI E1.11 - USITT DMX512-A
Have any exceptions been required? (Yes means that the implementation does not comply with the standard)	No [] Yes []
Date of Statement	

11.2.3 Major classifications

Item	Feature	Clause	Comment	Supported
GT	Transmitter - Grounded	4.6	Preferred Method; If No, compliance with Annex A required	No [] Yes [] N/A []
IR	Receiver - Isolated	4.6	Preferred Method; If No, compliance with Annex A required	No [] Yes [] N/A []
CHAS	Power Isolated	5.2		No [] Yes [] N/A []
EF	Enhanced Functionality (Secondary Data Link)	4.8.1	Active Method; If Yes, compliance with Annex B required	No [] Yes [] N/A []
PD	Processing Device			No [] Yes [] N/A []
HPL	Higher Protection Levels		If Yes, compliance with Annex C required	No [] Yes [] N/A []
XLR5	Pluggable Connection - 5-Pin XLR	7.1	If No, compliance with 7.1.2 Required	No [] Yes [] N/A []
NCC	Connection NCC	7.1.2		No [] Yes [] N/A []
HW	Connection - Hard Wire	7.2		No [] Yes [] N/A []
ASC	Generates Alternate START Code(s)		If Yes, compliance with Annexes D & E also required	No [] Yes [] N/A []

11.3 PICS tables for USITT DMX512-A

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
PL1	Comply with EIA-485-A	4.1	Compliant	M	No [] Yes []
PL2	Transmitter - line driver	4.3	Use a single active differential driver	M	No [] Yes [] N/A []
PL3	Receiver - line receivers	4.3	Use differential receiver	M	No [] Yes [] N/A []
PL4	Secondary data link	4.8.2	Passive use with input & output ports	M	No [] Yes [] N/A []
PL5	Secondary data link - containing three or more DMX512 ports	4.8.2	May provide a passive link between only two of the ports	M	No [] Yes [] N/A []
PL6	Secondary data link ports	4.8.2	Declared per clause 10	M	No [] Yes [] N/A []
PL7	Secondary data link ports	4.8.2	Marked on product	M	No [] Yes [] N/A []
PL8	Data link termination	4.9	Data link termination	M	No [] Yes [] N/A []
PL9	Data link termination	4.9	Operate in accordance with the stated manufacturer's specification when the data link is terminated	M	No [] Yes [] N/A []
PL10	Data link termination	4.9	120 Ohms + 5% / -10% between data- & data+	M	No [] Yes [] N/A []
PL11	Data link termination - transmitter on one end	4.9	Far end terminated	M	No [] Yes [] N/A []
PL12	Data link termination - transmitter not connected at one end	4.9	Both ends terminated	M	No [] Yes [] N/A []
PL13	Internal data link termination	4.9		M	No [] Yes []
PL14	Internal data link termination - Marking	4.9	If implemented, marking per clause 10	M	No [] Yes [] N/A []
PL15	Unpowered devices	4.10	No degradation of DMX512 transmission system performance	M	No [] Yes []
PL16	Unpowered devices	4.10	Not materially lower the impedance they present to the data link	M	No [] Yes []
OC	Operating characteristics	5.1	Measured at the ports	M	No [] Yes [] N/A []

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
GT1	Grounded transmitter	5.4	Provision for connection to protective earth	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
GT2	Grounded transmitter	5.4 / Figure 1	Any resistance (A) between data link common and zero volt supply (circuit common) of the transmitter circuit is less than 100 ohms	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Value if Yes: _____
GT3	Grounded transmitter	5.4 / Figure 1	Any resistance (B) between data link common and chassis is less than 20 ohms and is preferably zero ohms.	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Value if Yes: _____
GT4	Grounded transmitter	5.4	Transmitter not marked ISOLATED even though electrical Isolation may be used in the transmitter	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
DTT	Disallowed transmitter topology	5.5	Transmitter Not per figure 2	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
IR1	Isolated receiver	5.7	Any resistance (B) greater than 22 Mohm at 42 VDC with respect to chassis and any other signal inputs or outputs, and with respect to other ground referenced electronics	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Value if Yes: _____
IR2	Isolated receiver	5.7	Any resistance (B) greater than 22 Mohm at 42 VDC with respect to protective ground (where fitted)	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Value if Yes: _____
IR3	Isolated receiver	5.7 / Figure 3	Any resistance (A) between data link common and zero volt supply (circuit common) of the receiver circuit is less than 100 ohms	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Value if Yes: _____
DRT	Disallowed receiver topology	5.8	Transmitter not per figure 4	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
LDT	Transmission line loading	5.10	Transmitter capable of driving 32 unit loads	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
LDR1	Transmission line loading	5.10	Receiver presents unit load ≤ 1	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
LDR2	Transmission line loading	5.10	A receiver biased to any voltage from -7 to +12 volts does not present a capacitive load to the line of more than 125pf per unit load	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
LDM	Unit load marking	5.10	Optional declared or marked value is the greater of either the DC unit load determined by EIA-485-A clause 4.1 or the unit load as determined by the capacitive loading	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Value if Yes: _____
LDMC	Capacitive unit load marking	5.10	Optional declared or marked value - capacitive is also declared	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Value if Yes: _____
ESD	Electro static discharge protection	6.2	Withstand 4 kV ESD for contact discharge and 8 kV ESD for air discharge in accordance with IEC 61000-4-2 or other local regulations	M	No <input type="checkbox"/> Yes <input type="checkbox"/>
CM1	Connection method - plug-gable controller	7.1	Female	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
CM2	Connection method - plug-gable receiver	7.1	Male	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
CM3	Connection method - loop through	7.1	Male Input / Female Output convention	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
NCC0	Not connector compatible allowed	7.1.2	XLR-5 not physically possible	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Connector type used if Yes: _____
NCC1	Not connector compatible provisions	7.1.2-1	The alternate connector is not any type of XLR connector	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
NCC2	Not connector compatible provisions	7.1.2-2	The alternate connector is not any type of IEC 60603-7 8-position modular connector (RJ45) - if No, compliance with 7.3 required	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
NCC3	Not connector compatible provisions	7.1.2-3	Declared and marked NCC	M	No [] Yes [] N/A []
NCC4	Not connector compatible provisions	7.1.2-4	Adapter to XLR-5 provided	M	No [] Yes [] N/A []
NCC5	Not connector compatible provisions	7.1.2-5	EF and ground / isolation declaration made	M	No [] Yes [] N/A []
FIC1	Fixed installation connections with XLR	7.2	Not other than XLR-5	M	No [] Yes [] N/A []
FIC2	Fixed installation connections with XLR-5	7.2	If XLR-5, comply with 7.1, 7.1.1, and 7.1.2	M	No [] Yes [] N/A []
FIC3	Fixed installation connections	7.2	EF Markings per clause 10 and Annex B	M	No [] Yes [] N/A []
RJ45U	IEC 60603-7 8-position modular connectors - use	7.3	Use limited to connections part of a fixed installation not intended for regular connection and disconnection	M	No [] Yes [] N/A []
RJ45A	IEC 60603-7 8-position modular connectors - access	7.3	Not normally accessible except to qualified, authorized users, nor intended for regular connection and disconnection	M	No [] Yes [] N/A []
RJ45C	IEC 60603-7 8-position modular connectors wiring	7.3 / Table 4	Connections per Table 4	M	No [] Yes [] N/A []
DF1	Data format	8.1	Asynchronous and Sequential (Slot 0 - Last Slot - Max 513 Total)	M	No [] Yes []
DF2	Data format	8.1	Reset Sequence Transmitted	M	No [] Yes []
DF3	Data format	8.1	Valid slot value 0 to 255 decimal	M	No [] Yes []
DCD1	Dimmer class data	8.5.2	Valid dimmer levels are 0 to 255 decimal (00h to FFh)	M	No [] Yes [] N/A []
DCD2	Dimmer class data	8.5.2	Value 0 = dimmer output of OFF or minimum; Value 255 = output of FULL	M	No [] Yes [] N/A []

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
DCD3	Dimmer class data	8.5.2	Dimmer responds to increasing the DMX512 slot value from 0 to 255 by fading from its minimum level (off) to its maximum level (full)	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
ASC1	Alternate START codes	8.5.3	Registered ASC Used for Proprietary Information	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
ASC2	Alternate START codes interleaved with NULL START codes-refresh	8.5.3.1	NULL START Code packet sent at least once per second	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/>
ASC3	Alternate START codes - timing differences	8.5.3.2	Use one of two methods	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Method if Yes: _____
ASC4	Alternate START codes - inline processing	8.5.3.3	Declare how ASCs are processed	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Method if Yes: _____
ASC5	Alternate START codes - inline processing	8.5.3.3	Not pass some ASC packets and block others packets containing the same Alternate START code unless doing so as part of a stated processing algorithm	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Method if Yes: _____
ASC6	Alternate START codes	8.5.3.3	DMX512 in-line repeating transmitters do not pass some packets with a particular Alternate START code while blocking other packets containing the same Alternate START code unless doing so as part of a stated processing algorithm	M	No <input type="checkbox"/> Yes <input type="checkbox"/> N/A <input type="checkbox"/> Method if Yes: _____
SCP	Processing of START codes	8.5.4	Receivers other than in-line processing devices process the START code and differentiate between NSC and ASC packets	M	No <input type="checkbox"/> Yes <input type="checkbox"/>
SCI	Ignoring of START codes	8.5.4	Not ignore START codes by assuming that all packets received are NSC packets	M	No <input type="checkbox"/> Yes <input type="checkbox"/>

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
DLS1	Data link slots support	8.6	Data link supports up to 513 slots	M	No [] Yes []
DLS2	Multiple data links	8.6	Multiple links where more than 513 slots (512 data slots) are required	M	No [] Yes []
DLS3	Data link slots minimum	8.7	Data Link with less than 513 slots supported	M	No [] Yes []
LSBS1	Line state between data slots - value	8.8	Maintain Marking State	M	No [] Yes []
LSBP2	Line state between packets	8.9	The line remains in a marking state throughout during MBB	M	No [] Yes []
LSBP3	Line state between packets	8.9	Transmitter does not produce multiple BREAKS between data packets	M	No [] Yes []
LSBP4	Multiple BREAKS	8.9	Receiver capable of recovering from multiple BREAKS (data line errors)	M	No [] Yes []
TD	Timing Diagram	8.11 / Figure 5	DMX512 frames & packets format	M	No [] Yes []
TT	Transmitter timing	8.11 / Table 6		M	No [] Yes []
RT	Receiver timing	8.11 / Table 7		M	No [] Yes []
IFS1	Improperly framed slots	9.1	Check first stop bit	M	No [] Yes []
IFS2	Improperly framed slots	9.1	Discard data slot with missing stop bit and all subsequent data slots in packet	M	No [] Yes []
LDH	Loss of data handling	9.2 / 10.5.3	Declare loss of data handling procedure	M	No [] Yes []
RPM	Receiver performance at maximum data refresh	9.3	Receiver operates correctly while receiving any valid data packet	M	No [] Yes []
PPL	Packet processing latency	9.4 / 10.5.4	Processing latency declared	M	No [] Yes []
EFP1	EF port marking	10.2 / 10.5.1 / 10.5.2	Enhanced functionality ports marked	M	No [] Yes [] N/A []

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
EFP2	EF port marking declaration	10.2 / 10.5.1 / 10.5.2	Enhanced functionality ports declared	M	No [] Yes [] N/A []
PM1	Port marking - signal designation	10.2 / Table 8 / 10.5.2	Signal identification	M	No [] Yes [] N/A []
PM2	Port marking - signal designation abbreviations	10.2 / Table 8 / 10.5.2	Signal identification abbreviations	M	No [] Yes [] N/A []
DLTM	Data link termination marking	10.3	Switchable termination marked	M	No [] Yes [] N/A []
GIM	Ground / Isolation marking	10.4 / Table 9 / 10.5.1 / 10.5.2	Grounded or isolated port marked per Table 9	M	No [] Yes [] N/A []
NSC1	NULL START code functionality	10.5.5	Transmitters declare full range of slot values	M	No [] Yes [] N/A []
NSC2	NULL START code functionality	10.5.5	Receivers declare response to NSC slot values	M	No [] Yes [] N/A []
SFD	Slot footprint declaration	10.5.6	Receivers declare slot footprint	M	No [] Yes [] N/A []

Annex A - Non Preferred (Alternate) topologies (Normative)

A1 Isolated transmitters

This standard permits the use of isolated transmitter ports. They are used in legacy systems where grounded receivers are used and large Common Mode voltages are expected.

Figure A1 illustrates an isolated transmitter port. To be considered an isolated transmitter, any signal pin or contact of the DMX512 output shall present a resistance (B) greater than 22 Mohm at 42 VDC with respect to Chassis, and with respect to Protective Ground (where fitted). If there are any other signal inputs or outputs, any signal pin or contact of the DMX512 output shall present a resistance greater than 22 Mohm at 42 VDC with respect to these inputs or outputs. The power supply for the transmitter and any directly connected electronics shall be isolated from earth ground and any other grounded referenced electronics to levels at least as high as those required of the output. Any resistance (A) between Data Link Common's pin or contact and zero volt supply (circuit common) of the transmitter circuit shall be less than 100 ohms. There may be a capacitance (not shown) between Data Link Common and chassis for the purpose of Radio Frequency bypass.

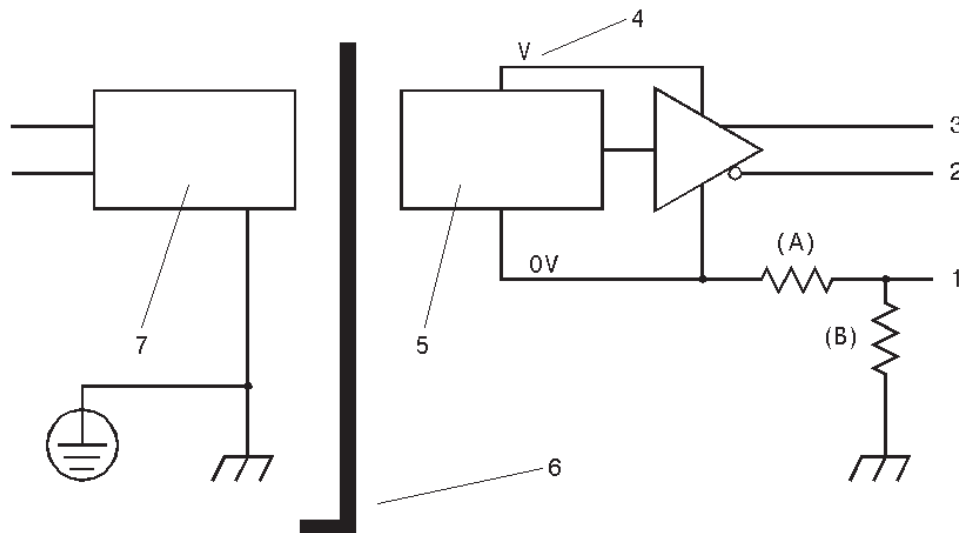


Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- 4 - V (+ or -)
- 5 - Other Isolated Electronics
- 6 - Isolation Barrier
- 7 - Non-Isolated Electronics (optional)
- A - Optional Resistance (see text)
- B - Resistance (see text)

Figure A1 - Isolated Transmitter

Table A1 - Isolated Transmitter Characteristics

Connection	Limit (measured at port pin or contact)	Comment
Data 1- to Data Link Common or Data 1+ to Data Link Common	+/- 6 V	Common Mode
Data 2- to Data Link Common or Data 2+ to Data Link Common	+/- 6 V	Enhanced Function Devices Only
Data Link Common to Chassis	$\geq 22\text{M ohm @ } 42\text{ VDC}$	
Data 1- to Chassis or Data 1+ to Chassis	$\geq 22\text{M ohm @ } 42\text{ VDC}$	
Data 2- to Chassis or Data 2+ to Chassis	$\geq 22\text{M ohm @ } 42\text{ VDC}$	Enhanced Function Devices Only
Data 1- to Data 1+	+/- 6 V	
Data 2- to Data 2+	+/- 6 V	Enhanced Function Devices Only
Any Port Pin or Contact to Chassis	30 VAC / 42 VDC	

A DMX512 device may have any number of isolated transmitter ports.

Adherence to this topology allows a DMX512 transmitter connector to be marked ISO or

A2 Non-isolated receivers

While not the preferred topology, this non-isolated topology exhibits considerable improvement in common mode fault tolerance compared to other grounded or non-isolated topologies. In this topology there shall be a resistance (B) of 100 ohms between Data Link Common's pin or contact and chassis. This resistance shall be able to safely dissipate two watts. No other connection between data link common and chassis is permitted. Common Mode voltage present between the chassis of this device and the chassis of any other device connected to the DMX512 shield will cause a current flow. This current will cause a voltage drop in the resistance (B). This voltage drop effectively decreases the Common Mode voltage at this receiver. Any resistance (A) between Data Link Common's pin or contact and zero volt supply (circuit common) of the receiver circuit shall be less than 100 ohms.

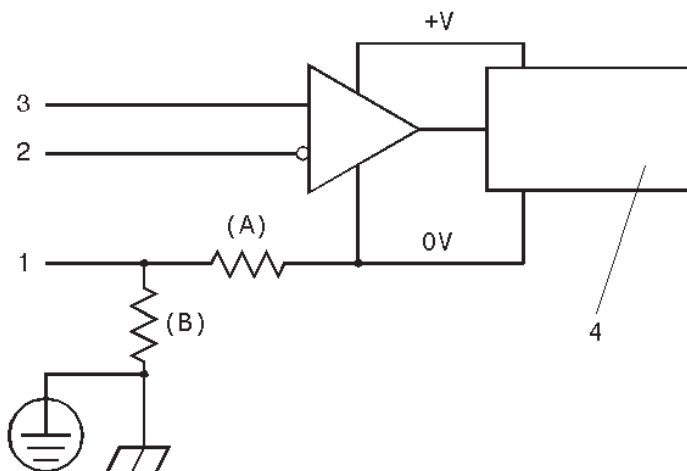


Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- 4 - Other Electronics
- A - Optional Resistance (see text)
- B - Optional Resistance (see text)

Figure A2 - Non-Isolated Receiver

Since 0 V is not directly referenced to Chassis, local product safety standards may restrict choice of power supply (e.g., use of a Class 2 supply).

A DMX512 receiver may have any number of non-isolated receiver ports. Where multiple ports are implemented, the total parallel resistance (B) shall be 100 ohms.

Table A2 - Non-Isolated Receiver Characteristics

Connection	Limit (measured at port pin or contact)	Comment
Data 1- to Data Link Common or Data 1+ to Data Link Common	+12 /- 7 VDC	Common Mode
Data 2- to Data Link Common or Data 2+ to Data Link Common	+12 /- 7 VDC	Enhanced Function Devices Only
Data Link Common to Chassis	100 ohms	-Note 1-
Data 1- to Chassis or Data 1+ to Chassis	+12 /- 7 VDC	
Data 2- to Chassis or Data 2+ to Chassis	+12 /- 7 VDC	Enhanced Function Devices Only
Data 1- to Data 1+	+/- 6 V	
Data 2- to Data 2+	+/- 6 V	Enhanced Function Devices Only
Any Port Pin or Contact to Chassis	N/A	

Note 1: This cannot be characterized in terms of voltage.

Manufacturers shall be permitted to fit a resistance of 100 ohms +/-20% between Chassis and Data Link Common for the purpose of limiting current in the shield due to small differential ground potentials. This method provides for reduction of Common Mode voltage at the line receiver.

The only output that shall be permitted to be directly connected to this topology is a single passive DMX512 loop through port. No other inputs or outputs are allowed with this topology unless they meet the requirements of clause A4.

A3 Grounded receivers

The topology of figure A3 is allowed for the construction of entry level receivers where the cost of isolation might prove an untenable burden. It may be used by manufacturers of receivers who, for reasons beyond the scope of this Standard, require a direct link between data link common and protective earth. It is not a recommended practice and requires special marking on the product and special explanatory text in all manuals.

This topology is characterized by the direct connection of the shield (Data Link Common) to chassis and protective earth. Therefore, devices employing ground referenced receivers shall be provided with a connection to protective earth. Any resistance (A) between Data Link Common's pin or contact and zero volt supply (circuit common) of the transmitter circuit shall be less than 100 ohms.

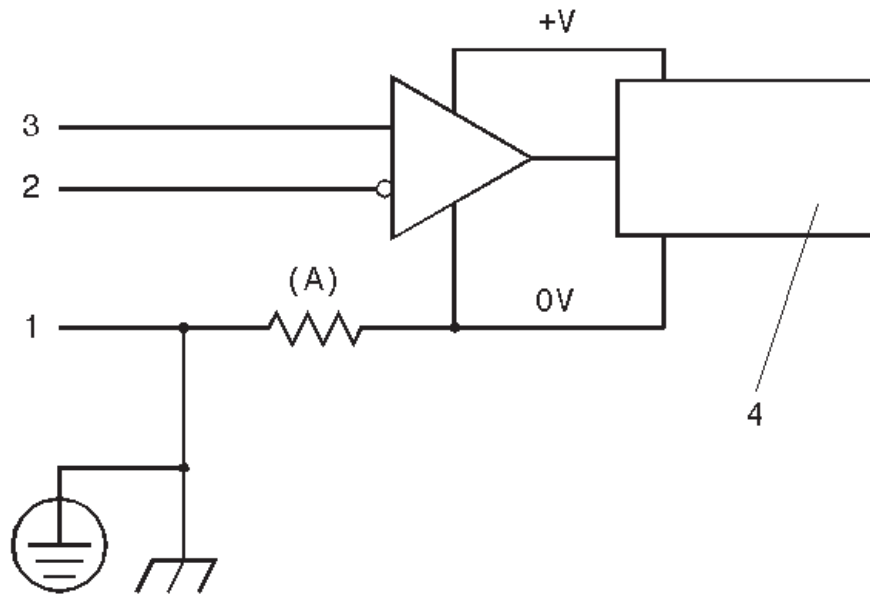


Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- 4 - Other Electronics
- A - Optional Resistance (see text)

Figure A3 - Grounded Receiver

Ports using this topology shall be marked using the standard symbol representation as defined in clause 10.5 and be declared.

A4 Earth grounding of data link common for floating devices

Figure A4 illustrates a floating topology. Floating is both an input and an output topology and is an additional allowable topology. It is often confused with the isolated DMX512 topology. As with the isolated topology, any input or output signal pin or contact shall present a resistance (B) greater than 22 Mohm at 42 VDC with respect to Chassis, and with respect to Protective Ground (where fitted). It differs from an isolated topology in that there is no requirement of isolation between DMX512 inputs and outputs. The resistance between Data Link Common's pin or contact of the DMX512 input and Data Link Common's pin or contact of the DMX512 output shall be less than 0.2 ohms. There may be a capacitance (not shown) between Data Link Common and chassis for the purpose of Radio Frequency bypass. Any resistance (A) between any Data Link Common's pin or contact and the zero volt supply (circuit common) of the transmitter and receiver circuits shall be less than 100 ohms.

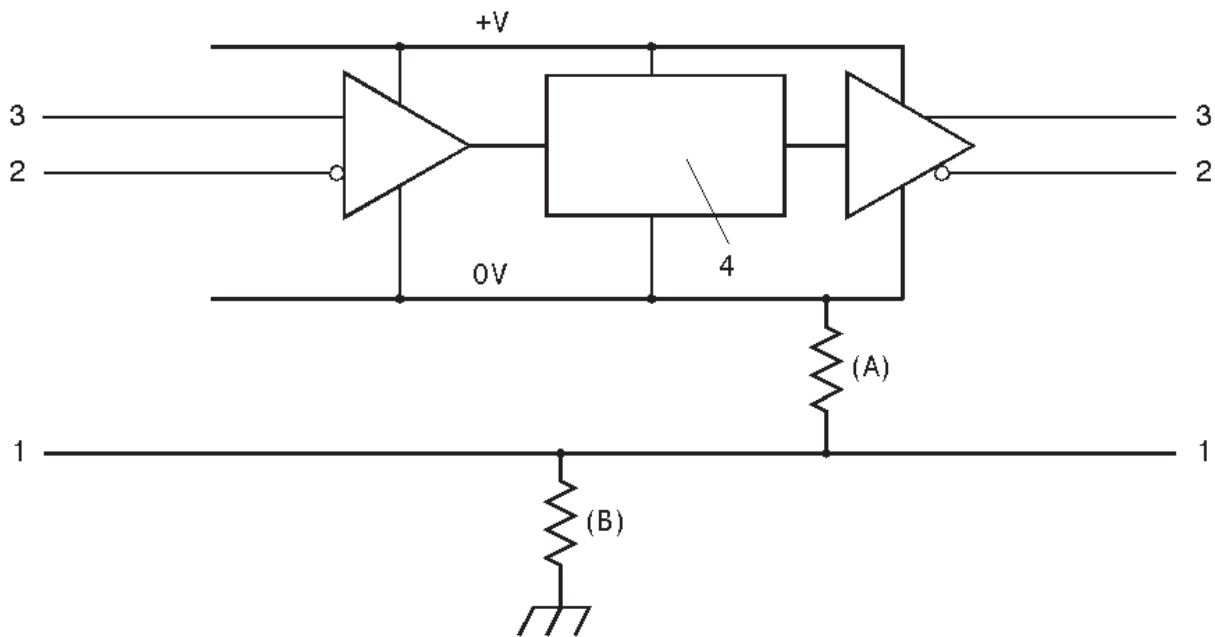


Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- 4 - Other Electronics
- A - Optional resistance (see text)
- B - Optional resistance (see text)

Figure A4 - DMX512 Device, Floating

The grounding of a device using this topology is determined by the connected devices. For that reason this topology shall not be used for devices that provide ground referenced non-DMX512 input or output ports unless those ports are isolated from DMX512 lines by an impedance of at least 22 Mohms.

Both input and output ports of a floating device shall be marked FLT, FLOAT, or FLOATING. A device may have only one floating DMX512 receiver port.

Annex B - (Normative) - Enhanced DMX512

B1 General

The original and 1990 versions of USITT DMX512 called out an “Optional Second Data Link.” There was no detailed guidance for its use. The majority of legacy systems did not use the second data link at all. Many uses of the second data link have been implemented over the years. While many of these were reasonable, a few uses clearly were not compliant with EIA-485-A. These uses vary in both their electrical requirements and in the data protocol used. One of the purposes of this standard is to regularize the use of the second data link. It is no longer possible to select a single implementation and forbid all others. However, not all historical uses will be allowed to continue.

The network topologies needed to support Enhanced Functions are identified by an EF number. This edition of the standard supports four different EF topologies.





The introduction of standard EF topologies in no way changes backward compatibility of primary data link functions. In all cases, regardless of the EF topology, all DMX512 transmitters and receivers shall be able to interchange DMX512 data on the primary data link. All DMX512 devices shall be able to be connected without damage. All DMX512 devices shall not be damaged by connection to compliant legacy devices.

A DMX512 controlled device capable of returning status information is referred to as a responder in this standard. For the purposes of this standard, responders are not considered to fall under the category of transmitting devices.

B2 Summary of Enhanced Function Topologies

The Enhanced Function topologies are summarized in table B1. The data format on primary data link (Data 1- and Data 1+) shall adhere to the DMX512-A Standard.

Table B1 - Enhanced Function Topologies

EF#	Symbol (Optional)	Description	Comment
1		half duplex EIA-485-A signals on primary data link; return signals on primary data link controlled by a registered Alternate START Code; no functionality on secondary data link	
2		unidirectional DMX512 data on primary data link (EIA-485-A signals) and return EIA-485-A signals on secondary data link	
3		unidirectional DMX512 data on primary data link (EIA-485-A signals) and half duplex EIA-485-A signals on secondary data link	refer to manufacturers instructions
4		half duplex EIA-485-A signals on both pairs; return signal on primary data link controlled by a registered Alternate START Code	

Note: References to unidirectional data are with respect to a transmitting device.

Different physical DMX512 ports on a product may be of different EF topologies and shall be declared and marked as required in clause 10.

B3 Identification of data protocols for Enhanced DMX512

The EF topology specifies the electrical and operating mode for the data links. It does not detail the data structure or protocol. The method of identifying data structures or protocols using the defined EF numbers is by way of an additional registered number placed after the EF number, separated from the EF number by a period. The registry for these designations will be maintained by the Secretariat for the ANSI E1 Accredited Standards Committee – ESTA (see Annexes D and E). Numbers ending in zero are reserved for future development of the standard (e.g., EF1.0, EF1.10, EF2.20).

B3.1 EF1 Half duplex DMX512 - Bi-directional use of the primary data link

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 signal packets as well as return data controlled by the use of Alternate START Code packets. There is no specific requirement that EF1 devices use NULL START Code packets for any of their functions – that requirement is defined by the specific EF protocol.

An EF1 system shall use polled responses on the primary data link, such that a receiving device can not transmit a packet unless instructed to by an appropriate Alternate START Code packet from the controller. All response packets shall begin with an Alternate START Code. Once configured, any functions using EF1 shall use collision free data protocols.

All packets in an EF1 system shall follow the normal timing and limits for a DMX512 packet. There may be additional timing limits imposed by the chosen protocol. Any additional timing limits are beyond the scope of this standard.

B3.1.1 Bi-directional distribution amplifiers for EF1

Systems using EF1 topologies require bi-directional distribution/return data combiners. These combiners shall control the direction of data flow on the primary data link. The design of such a bi-directional distribution/return data combiner requires detailed understanding of the protocol timings. Therefore, the design of bi-directional distribution/return data combiners for EF1 without knowledge of the protocol to be used is meaningless. The requirements for such designs are beyond the scope of this standard.

B3.1.2 Termination of EF1 system

Termination of EF1 primary data links shall conform to the electrical requirements of clause 4.9. Additionally, EF1 systems shall terminate each end of the primary data link. Therefore, EF1 controllers and other transmitting devices shall provide a means to terminate the primary data link. Such terminations may also require specific line biasing or other methods to ensure integrity of returned data that are beyond the scope of this standard.

B3.2 Full duplex DMX512

The EF2 topology uses the second data link to provide a data path to return information from controlled devices. The data on second data link shall be unidirectional flowing from the controlled

devices to the controller. This data link will operate in multi transmitter multipoint mode. In this mode the primary and secondary links comprise a full duplex data link.

Each responder shall have an EIA-485-A transmitter for the second data link with a driver enable control. The driver enable control shall be driven so that each responder can control the state of the return data link while transmitting its response bytes. Once configured only one responder shall be enabled at once and there shall be at least one bit time between one responder going inactive and the next going active. In general, the line driver can be enabled one bit time prior to transmission, and disabled one bit time after the last bit has been sent.

Once configured, any functions using EF2 shall use collision free data protocols. EF2 protocols shall also be structured to allow the use of data distribution amplifiers that meet the requirements of clause B3.2.2.

B3.2.1 Wiring of EF2 DMX512 ports

EF2 responders having two DMX512 ports for receive and transmit shall provide a direct passive link for Data 2- and Data 2+ between those ports (clause 4.8.2). The responder transmitter shall be connected to this passive link. Devices having three or more DMX512 ports shall wire the additional ports in a manner appropriate for the device's functionality. The manner in which these ports are wired shall be clearly detailed in the product's manual. Devices having three or more ports shall have two ports wired as a loop through with an attached responder transmitter.

B3.2.2 Bi-directional distribution amplifiers for EF2

Systems using EF2 topologies require bi-directional distribution amplifiers/return data combiners. The DMX512 primary data link may be split and separately buffered as in standard DMX512 buffers. Return data receivers shall be "wire-OR" connected within a unit. This combined received data signal is used to drive back to the return data monitor.

The bi-directional distribution amplifier shall not perform any processing on the data, since some response protocols depend upon the relationship with outbound DMX512 to synchronize the return data.

B3.2.3 Termination of EF2 data links

EF2 systems shall terminate the primary data link following the requirements of clause 4.9. They shall also terminate the secondary data link to conform to the electrical requirements of 4.9 except that both ends of the secondary data link shall be terminated. Therefore, EF2 controllers and other transmitting devices shall provide a means to terminate the secondary data link. Such terminations may also require specific line biasing or other methods to ensure integrity of returned data that are beyond the scope of this standard.

B3.3 EF3 Half duplex on the second data link

Systems that send data in both directions on the secondary data link are classified as EF3. These systems shall use the primary data link to send a standard unidirectional DMX512 signal.

B3.3.1 Bi-directional distribution amplifiers for EF3

Systems using EF3 topologies require bi-directional distribution/return data combiners. These combiners shall control the direction of data flow on the secondary data link. The design of such a bi-directional distribution/return data combiners requires detailed understanding of the protocol timings. Therefore, the design of bi-directional distribution/return data combiners for EF3 without knowledge of the protocol to be used is meaningless. The requirements for such designs are beyond the scope of this standard.

B3.3.2 Termination of EF3 data links

EF3 systems shall terminate the primary data link following the requirements of clause 4.9. They shall also terminate the secondary data link to conform to the electrical requirements of 4.9 except that both ends of the secondary data link shall be terminated. Therefore, EF3 controllers and other transmitting devices shall provide a means to terminate the secondary data link. Such terminations may also require specific line biasing or other methods to ensure integrity of returned data that are beyond the scope of this standard.

B3.4 EF3 Protocols that use the primary and the secondary data links in ways not covered above

Systems that use the primary and the secondary data links to send data in both directions are classified as EF4. Both data links shall comply with the requirements of clauses B3.1 (EF1) and fall within the scope of EIA-485-A except that the use of the secondary data link is optional.

B3.5 Additional electrical requirements

The details of a particular protocol using an EF topology may necessitate additional electrical requirements. These requirements are beyond the scope of this document.

Annex C - (Normative) - Higher Protection Levels - “DMX512-A Protected”

Some manufacturers feel it is prudent to employ higher levels of protection on their DMX512 ports than is specified in EIA-485-A. Many semiconductor manufacturers have recognized the need for higher protection and have produced “fault protected transceivers.” The tables below indicate the minimum requirements for a protected DMX512 port. If a manufacturer’s DMX512 port meets these requirements, the port can be declared as “DMX512-A Protected”.

Table C1 - Transmitter Protection

Connection	Minimum Protection Limit		
Data 1- to Data Link Common or Data 1+ to Data Link Common	30 VAC / \pm 42 VDC		
Data 2- to Data Link Common or Data 2+ to Data Link Common	30 VAC / \pm 42 VDC		
Data 1- to Data 1+	30 VAC / \pm 42 VDC		
Data 2- to Data 2+	30 VAC / \pm 42 VDC		
	Ground Referenced	Isolated	Floating
Data Link Common to Chassis	< 0.2 ohms	N/A	N/A
Any other non Data Link Common Port Pin or contact to Chassis	30 VAC / \pm 42 VDC	N/A	N/A
Any Port Pin or Contact to Chassis	N/A	30 VAC / \pm 42 VDC	30 VAC / \pm 42 VDC

Table C2 - Receiver Protection

Connection	Minimum Protection Limit		
Data 1- to Data Link Common or Data 1+ to Data Link Common	30 VAC / \pm 42 VDC		
Data 2- to Data Link Common or Data 2+ to Data Link Common	30 VAC / \pm 42 VDC		
Data 1- to Data 1+	30 VAC / \pm 42 VDC		
Data 2- to Data 2+	30 VAC / \pm 42 VDC		
	Non-Isolated	Isolated	Floating
Data Link Common to Chassis	-note 1-	N/A	N/A
Any other non Data Link Common Port Pin or contact to Chassis	30 VAC / \pm 42 VDC	N/A	N/A
Any Port Pin or Contact to Chassis	N/A	30 VAC / \pm 42 VDC 100Mohm at 50 VDC	30 VAC / \pm 42 VDC \geq 22Mohm at 50 VDC

Note 1: This cannot be characterized in terms of voltage. Clause A2 (Receiver Characteristics) allows manufacturers to fit a resistance between chassis and Data Link Common for the purpose of limiting the current in the shield due to small ground differentials. Any such resistance shall survive continuous connection to voltages within the EIA-485 Common Mode range of -7/+12 VDC.

Annex D - (Normative) - Reserved Alternate START Codes

D1 Reserved Alternate START Codes

Several Alternate START Codes are reserved for special purposes or for future development of the Standard. No equipment shall be manufactured that generates Alternate START Codes 92 - A9 (146 - 169 decimal) or AB - CD (171 - 205 decimal) until their use is defined by the Standard or by the E1 Accredited Standards Committee. Manufacturers shall not advertise or sell products or devices that use Alternate START Codes F0 - F7 (240 - 247 decimal).

Table D1 - Reserved Alternate START Codes

Alternate START Codes			
Hexadecimal	Decimal	Purpose	Note
17h	23	Text Packet	see Annex Clause D2 for implementation
55h	85	Test Packet	see Annex Clause D3 for implementation
90h	144	90h is reserved for future expansion	
91h	145	91h followed by a 2 byte Manufacturer ID field is reserved for Manufacturer/Organization specific use, transmitted byte order is MSB, LSB. The next byte after the Manufacturer's ID would normally be a manufacturer's sub-code	The Manufacturer ID serves as an identifier that the data following in that packet is proprietary to that entity and should be ignored by all others
92h - A9h	146 - 169	possible future revisions of this Standard	Use not currently permitted
ABh - CDh	171 - 205	possible future revisions of this Standard	Use not currently permitted
CFh	207	System Information Packet	see Annex Clause D4 for implementation
F0h - F7h	240 - 247	prototyping/experimental use while the manufacturer/organization is waiting for their registered Alternate START Code to be assigned	Not permitted in shipping product

D2 ASC text packet

Alternate START Code 17h (23 decimal) shall designate a special packet of between 3 and 512 data slots. The purpose of the ASC text packet is to allow equipment to send diagnostic information formatted for display.

Slot allocation is as follows:

Slot 1: Page number of one of the possible 256 text pages.

Slot 2: Characters per Line. Indicates the number of characters per line that the transmitting device has used for the purposes of formatting the text. A slot value of zero indicates ignore this field.

Slots 3-512: Consecutive display characters in ASCII format. All characters are allowed and where a DMX512 text viewer is capable, it shall display the data using the ISO/IEC 646 standard character set. A slot value of zero shall terminate the ASCII string. Slots transmitted after this null terminator up to the reset sequence shall be ignored.

D3 ASC test packet

Alternate START Code 55h (85 decimal) shall designate a special test packet of 512 data slots, where all data slots carry the value 55h (85 decimal). Test packets shall be sent so that the time from the start of the Break until the stop bit of the 513th slot shall be no more than 25 milliseconds. When test packets are sent back to back, the Mark Before Break time shall be no more than 88 microseconds. The Break timing for test packets shall be greater than or equal to 88 microseconds, and less than or equal to 120 microseconds. The Mark After Break time shall be greater than or equal to 8 microseconds and less than or equal to 16 microseconds.

D4 System information packet (SIP) Alternate START Code

Alternate START Code CFh (207 decimal) is reserved for a System Information Packet (SIP). The SIP includes a method of sending checksum data relating to the previous NULL START Code packet on the data link and other control information. No other packet shall be sent between the NULL START Code packet and the SIP that carries its checksum.

D4.1 Application

Manufacturers of control consoles are encouraged to transmit SIPs, either as a background to normal processing or, in conjunction with the special test packet, as part of their suite of system test functions. One of the current problems with testing of DMX512 installations is that it must be done with static test packets – certain modes of testers cannot be used while a console is actually running the show, as by definition the DMX512 packets are varying as each cue runs. The interleaving of SIP's would allow some degree of live testing, particularly if one or more test packets were also sent applicable to the functionality of the receiving device.

Note: For systems requiring a more reliable link, manufacturers would have the option of following every normal packet with a SIP packet, although it is recognized that this would degrade data throughput. It could be used with systems that send packets of fewer than 512 DMX512 data slots or refresh data at less than the maximum rate.

D4.2 SIP format

The SIP Packet Length is 24 data slots in the format specified in Table D2. Receivers shall be required to accept SIPs of up to 255 data slots to allow for future expansion.

D4.3 SIP checksum pointer

Transmitting devices shall send a value of 24 in slot 1 that represents the length of SIP Packet in this version of the Standard. Receivers shall use the value received in slot 1 to establish the offset to the SIP checksum.

Table D2 - SIP Format

Slot	Definition	Re-fer to Clause
1	SIP Byte Count/SIP Checksum pointer (valid value is 24)	D4.3
2	Control Bit Field	D4.4
3	MSB of 16 bit additive Checksum of previous packet	D4.5
4	LSB of the 16 bit Checksum of previous packet	D4.5
5	SIP sequence number	D4.6
6	DMX512 universe number	D4.7
7	DMX512 processing level	D4.8
8	Version of Software sending this SIP	D4.9
9	Standard Packet Len MSB	
10	Standard Packet Len LSB	D4.10
11	Number of Packets transmitted by originating device since last SIP MSB	
12	Number of Packets transmitted by originating device since last SIP LSB	
13	Originating Device's Manufacturer ID MSB	D4.11
14	Originating Device's Manufacturer ID LSB	D4.12
15	Second Device's Manufacturer ID MSB	D4.13
16	Second Device's Manufacturer ID LSB	D4.13
17	3rd Device's Manufacturer ID MSB	D4.13
18	3rd Device's Manufacturer ID LSB	D4.13
19	4th Device's Manufacturer ID MSB	D4.13
20	4th Device's Manufacturer ID LSB	D4.13
21	5th Device's Manufacturer ID MSB	D4.13
22	5th Device's Manufacturer ID LSB	D4.13
23	<i>reserved for future use - transmit as 0</i>	
24 - (nn-1)	<i>reserved for future use</i>	D4.3
nn (max 255)	8-bit Additive Checksum of the SIP	D4.14

D4.4 Control bit field

d7	d6	d5	d4	d3	d2	d1	d0
reserved transmit as 0	reserved transmit as 0	reserved transmit as 0	reserved transmit as 0	reserved transmit as 0	reserved transmit as 0	previous packet bit set = 1	subsequent NULL packet hold control bit set=1

D4.4.1 Subsequent NULL packet hold control bit d0

Processing of the subsequent NULL packet hold Control Bit is optional. If implemented, when the subsequent NULL packet hold Control Bit is set (=1), the subsequent NULL START Code packet shall be held pending the reception of the next SIP for validation of the checksum. If a second NULL START Code packet is received without a preceding SIP, the receiver shall return to an immediate use mode.

D4.4.2 Previous packet bit d1

Set (= 1) if previous packet was an Alternate Start Code (ASC) packet.

D4.5 Checksums

16 bit ones complement additive checksum of all slots in the previous packet. The checksum includes the START Code.

D4.6 SIP Sequence number

A free running 8 bit counter identifying the SIP and incremented by a SIP generator by 01h on every subsequent SIP. This field may be checked to ensure that SIPs have not been missed.

D4.7 Originating universe

This slot indicates the (originating) DMX512 universe currently transmitted on this link. 00h is not used. Valid values are 01h - FFh (1 decimal - 255 decimal).

D4.8 DMX512 processing level

This slot indicates the level of post controller processing. Originating devices shall always transmit a value of 00h in this field. Processing devices such as merge units or any that regenerate or provide a media conversion (e.g., Ethernet to DMX512) facility and do not explicitly block ASC packets per clause D3 shall increment the value of this field by 01h. The content of this field indicates a level of process “hops” that data on the link has been subjected to relative to the originating transmitting device.

D4.9 Software version

00h	not implemented
01h - FFh	firmware version of last device

Note: This slot is for use by the manufacturer and may not correlate with any formally published release identifier.

D4.10 Packet lengths

This declares the standard length of packets for START Code 00, normally transmitted on this link.

Valid values are	0000h packet length not declared
	0001h - 0200h designates value of fixed packet length
	0201h - 7FFFh are reserved
	8000h Dynamic Packet, length not declared
	8001h - 8200h length of last dynamic packet
	8201h - FFFFh are reserved

D4.11 Number of packets

A 16 bit count of the number of packets transmitted by the originating device since last SIP was transmitted. This count should not increment past FFFFh.

D4.12 Manufacturer ID

Manufacturer ID will be the same 16 bit assignment as used for the Manufacturer's ID field used with Alternate START Code 91h (see Annex E - clause E1).

an ID == 0000h indicates that Manufacturer is not declared.

an ID == FFFFh indicates that Manufacturer has applied for, but not been granted, an ID and that this transmission originates from a product under development.

D4.13 Packet history

DMX512 Processing devices and media converters that process SIPs shall be required to insert their own Manufacturer's ID into the SIP packet. An originating device shall always send its Manufacturer's ID in SIP slots 13 and 14, with 0000h in slots 15, 16; 17, 18; 19, 20 and 21, 22. Subsequent processing devices shall insert their manufacturer's ID into the slots as indicated by the DMX512 processing level slot. A processing level of 01h corresponds to the second device, a processing level of 02h corresponds to the third device, and so on.

Note: This scheme allows for a packet processing history to be traced back though a complex installation of products.

D4.14 SIP Checksum

8 bit ones complement additive checksum of the SIP START Code (CFh) and all subsequent slots of SIP data.

Annex E - (Normative) - Alternate START Code, Manufacturer ID, and Enhanced Functionality Registration

E1 Alternate START code registration policy - 1 - 255 decimal (01 - FF hexadecimal)

Slot 0 of a DMX512 packet is the START Code. The value of this slot identifies intended use of data in the rest of the packet. The Standard provides for a non NULL or "Alternate" START Code. Where it is required to send proprietary information over a DMX512 data link, a packet starting with a registered Alternate START Code shall be used.

E2 Authorized use

The E1 Accredited Standards Committee or any organization that it authorizes may use an Alternate START Code to provide further extensions to the DMX512 Standard.

E3 Reserved Alternate START Codes

Several Alternate START Codes are reserved for special purposes or for future development of the Standard. See Annex D.

E4 Requests for registration of new START Codes

Any manufacturer or organization involved in the use of DMX512 may request that a START Code or Manufacturer ID be registered for their use. Although not encouraged, an Alternate START Code and Manufacturer ID may be registered for proprietary use. Requests shall be forwarded to the Secretariat for the ANSI E1 Accredited Standards Committee – ESTA. ESTA will attempt to honor such reasonable requests as described below.

E4.1 Number of Alternate START Codes per entity

No more than one Alternate START Code may be registered to any one manufacturer/organization. Manufacturers and Organizations with Alternate START Codes registered prior to the publication of this Standard may request one additional Alternate START Code.

E4.2 Selection of the Alternate START Code value and Manufacturer ID

The assignment of any particular numeric START Code or Manufacturer ID value to any particular entity is solely at the discretion of Accredited Standards Committee E1. Assignment depends on the availability of unused and unreserved START Codes and Manufacturer IDs.

E5 Requirement for registration of an EF protocol

No EF protocol intended for use between multiple Manufacturers will be registered as an EF protocol unless its basic structure is available to anyone by request and can be used freely by any manufacturer. The entire message structure for the protocol does not have to be made public, only the portions that are for public use. It is expected that some protocols may still have portions of the message structure that are reserved for proprietary use (i.e., vendor specific messages). Registration is at the discretion of Accredited Standards Committee E1.

E6 Document register

E6.1 Documentation for use of Alternate START Codes

The manufacturer/organization requesting registration of an Alternate START Code shall provide a 2-line description of the purpose of the Alternate START Code. They shall list the minimum and maximum number of slots, including the START Code, in any proposed packet. Any provided description (subject to editing) shall be included in the Register. This is not required for functions associated with a manufacturer specific ID under Alternate START Code 91h. It is recommended, but not required, for Alternate START Codes assigned prior to the adoption of this version of the Standard.

E6.2 Maintenance and publication

The ANSI E1 Accredited Standards Committee through its secretariat (ESTA) shall maintain a Register of Alternate START Codes and Manufacturer IDs. ESTA will publish the Registry as needed.

E6.3 Supplement documentation

If the manufacturer/organization wishes detailed documentation to be in the Public Domain, a note will be added to the Registry, but they will be responsible for such publication.

E7 Ownership

The DMX512 Standards are copyrighted. By registering a START Code or Manufacturer ID, no ownership rights are conferred to any third party. Alternate START Codes are registered to particular entities solely to allow for orderly management of the Standard. The registrant does not own the Alternate START Code or Manufacturer ID.

Annex F - (Informative) - Protocol Implementation Conformance Statement (PICS for Annexes A through E

F1 Introduction

The supplier of a protocol implementation that is claimed to conform to any of the Annexes for ANSI E1.11, shall complete the following Protocol Implementation Conformance Statement (PICS) for each applicable annex.

F2 Implementation identification

F2.1 Identification

Manufacturer	
Contact information for inquiries about the PICS (Note 1)	
Implementation Name(s) and Version(s) (Notes 1,2)	
Other information necessary for full identification – e.g., name(s) and version(s) for equipment and/or operating systems; System Name(s) (Note 3)	

Note 1: Required for all implementations

Note 2: The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).

Note 3: May be completed as appropriate in meeting the requirements for the identification.

F2.2 Protocol summary

Identification of the Protocol Annex	ANSI E1.11 – USITT DMX512-A [] Annex A - Non Preferred (alternate) Topologies [] Annex B - Enhanced DMX512 [] [] Annex C - Higher Protection Levels (DMX512-A Protected) [] Annex D - Reserved Alternate START Codes [] Annex E - Alternate START Code, Manufacturer ID, and Enhanced Functionality Registration			
Have any exceptions been required? (Yes means that the implementation does not comply with the standard.)	Annex A -	No []	Yes []	N/A []
	Annex B -	No []	Yes []	N/A []
	Annex C -	No []	Yes []	N/A []
	Annex D -	No []	Yes []	N/A []
	Annex E -	No []	Yes []	N/A []
Date of Statement				

F3 (PICS) tables for annexes A through E

F3.1 Annex A - Non preferred (alternate) topologies

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
IT1	Isolated Transmitter	Annex A	Transmitter is Isolated	O If Yes, compliance with Annex A1 required	No [] Yes [] N/A []
IT2	Isolated Transmitter	A1	presents a resistance (B) greater than 22 Mohm at 42 VDC with respect to Chassis and any other signal inputs or outputs, and with respect to other ground referenced electronics	M	No [] Yes [] N/A [] Value if Yes: _____
IT3	Isolated Transmitter	A1	presents a resistance (B) greater than 22 Mohm at 42 VDC with respect to Protective Ground (where fitted)	M	No [] Yes [] N/A [] Value if Yes: _____
IT4	Isolated Transmitter	A1 / Figure A1	Any resistance (A) between Data Link Common and zero volt supply (circuit common) of the transmitter circuit is less than 100 ohms	M	No [] Yes [] N/A [] Value if Yes: _____
IT5	Isolated Transmitter	A1 / Table A1	Isolated Receiver Characteristics	M	No [] Yes [] N/A []
FR1	Non-Isolated Receiver	A2 / Figure A2	resistance (B) of 100 ohms is between Data Link Common and chassis	O If Yes, compliance with Annex A2 required	No [] Yes [] N/A []
FR2	Non-Isolated Receiver	A2 / Figure A2	resistance (B) safely dissipates two watts	M	No [] Yes [] N/A []
FR3	Non-Isolated Receiver	A2 / Figure A2	no other connection between data link common and chassis	M	No [] Yes [] N/A []
FR4	Non-Isolated Receiver	A2 / Figure A2	resistance (A) between Data Link Common and zero volt supply (circuit common) of the receiver circuit is less than 100 ohms	O	No [] Yes [] N/A [] Value if Yes: _____
FR5	Non-Isolated Receiver - Multiple Ports	A2	when multiple ports are implemented, the total parallel resistance (B) is 100 ohms	O	No [] Yes [] N/A []

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
FR6	Non-Isolated Receiver	A2 / Table A2	Non-Isolated Receiver Characteristics	M	No [] Yes [] N/A []
FR7	Non-Isolated Receiver Optional Shield Current Limiting	A2	resistance of 100 ohms +/-20% between Chassis and Data Link Common	O	No [] Yes [] N/A []
FR8	Non-Isolated Receiver Permitted Output Connection	A2	the only output directly connected to this topology is a single passive DMX512 loop through port	M	No [] Yes [] N/A []
FR9	Non-Isolated Receiver Permitted Output Connection	A2	no other inputs or outputs are allowed with this topology unless they meet the requirements of clause A4	M	No [] Yes [] N/A []
GR1	Grounded Receiver	A3	receiver is grounded	O If Yes, compliance with Annex A3 required	No [] Yes [] N/A []
GR2	Chassis / Protective Earth Connection	A3	chassis and Data Link Common are connected to protective earth	M	No [] Yes []
GR3	Grounded Receiver	A3 / Figure A3	any resistance (A) between Data Link Common and zero volt supply (circuit common) of the transmitter circuit is less than 100 ohms	M	No [] Yes [] N/A [] Value if Yes: _____
GR4	Grounded Receiver Port Marking	A3 / 10.5	Grounded Receiver Ports marked per 10.5	M	No [] Yes []
FLT1	Floating Devices	A4	Device is Floating	O If Yes, compliance with Annex A4 required	No [] Yes [] N/A []

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
FLT2	Floating Devices	A4 / Figure A4	input or output presents a resistance (B) greater than 22 Mohm at 42 VDC with respect to Chassis and Protective Ground (where fitted) and there is no isolation between DMX512 inputs and outputs	M	No [] Yes [] N/A []
FLT3	Floating Devices Data Link Common	A4	resistance between Data Link Common of the DMX512 input and pin 1 of the DMX512 output is less than 0.2 ohms	M	No [] Yes [] N/A []
FLT4	Floating Devices	A4 / Figure A4	any resistance (A) between Data Link Common and zero volt supply (circuit common) of the transmitter circuit is less than 100 ohms	M	No [] Yes [] N/A [] Value if Yes: _____
FLT5	Floating Device non-DMX512 devices	A4	not be used for devices that provide ground referenced non-DMX512 input or output ports unless those ports are isolated from DMX512 lines by an impedance of at least 22 Mohms	M	No [] Yes [] N/A []
FLT6	Floating Device Marking	A4	Device is Marked FLT, FLOAT, or FLOATING	M	No [] Yes [] N/A []

F3.2 Annex B - Enhanced DMX512

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
EF	Enhanced Functionality	Annex B	Enhanced Functionality implemented	O If Yes, compliance with Annex B required	No [] Yes [] N/A []
EF1	Primary Data Link Functionality	B1	Interchange of data on Primary Data Link	M	No [] Yes []
EF2	Damage Withstand	B1	DMX512 devices connected without damage	M	No [] Yes []

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
EF3	Damage Withstand - Compliant Legacy	B1	DMX512 devices not damaged by connection to compliant legacy devices	M	No [] Yes []
EFPC	Port Data Format	B2 / Table B1	data format on primary data link compliant with USITT DMX512-A	M	No [] Yes []
EFTD	Enhanced Functionality Topology and Port Declaration	B2 / Clause 10	EF topologies on each port declared and marked per clause 10	M	No [] Yes []
EF1-1	EF1	B3.1	Primary Data Link uses both NSC packets and ASC Controlled Return Data	M	No [] Yes []
EF1-2	EF1 Polled Responses	B3.1	Polled responses used on the primary data link only when instructed to by an appropriate Alternate START Code packet from the controller	M	No [] Yes []
EF1-3	EF1 Polled Response Packet	B3.1	Response packets begin with an Alternate START Code	M	No [] Yes []
EF1-4	EF1 Collision Free Protocol	B3.1	Once configured, EF1 functions use collision free data protocols	M	No [] Yes []
EF1-5	EF1 Timings	B3.1	EF1 Packets follow the normal timing and limits of a DMX512 packet	M	No [] Yes []
EF1-6	EF1 Bi-directional Distribution Amplifiers	B3.1.1	Combiners control the direction of data flow on the primary data link	M	No [] Yes [] N/A []
EF1-7	EF1 Data Link termination	B3.1.2	Termination of EF1 primary data links conforms to clause 4.9	M	No [] Yes []
EF1-8	EF1 Data Link termination	B3.1.2	EF1 systems terminate each end of the primary data link	M	No [] Yes []
EF1-9	EF1 Data Link termination	B3.1.2	EF1 controllers and other transmitting devices provide a means to terminate the primary data link	M	No [] Yes []

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
EF2-1	EF2	B3.2	Data (return data) on second data link is unidirectional from the controlled devices to the controller	M	No [] Yes [] N/A[]
EF2-2	EF2 Responder	B3.2	Responder has an EIA-485-A transmitter for the second data link with a driver enable control	M	No [] Yes []
EF2-3	EF2 Responder	B3.2	Each responder controls the state of the return data link while transmitting its response bytes	M	No [] Yes []
EF2-4	EF2 Responder	B3.2	Once configured only one responder is enabled at one time	M	No [] Yes []
EF2-5	EF2 Responder	B3.2	At least one bit time exists between one responder going inactive and the next going active	M	No [] Yes []
EF2-6	EF2 Protocol	B3.2	EF2 functions use collision free data protocols, structured to allow the use of data distribution amplifiers that meet the requirements of clause B3.2.2	M	No [] Yes []
EF2-7	Two Port Receive & Transmit EF2	B3.2.1 / 4.8.2	Device Wiring DMX512 ports for receive and transmit provide a direct passive link on secondary data link between those ports (clause 4.8.2)	M	No [] Yes [] N/A[]
EF2-8	Two Port Receive & Transmit EF2 Devices	B3.2.1	Responder transmitter be connected to passive link	M	No [] Yes []
EF2-9	Three or more Port Receive & Transmit EF2 Devices	B3.2.1	Devices having three or more DMX512 additional ports wired in a manner appropriate for the device's functionality	M	No [] Yes [] N/A[]

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
EF2-10	Three or more Port Receive & Transmit EF2 Devices	B3.2.1	The manner in which these ports are wired clearly detailed in the product's manual	M	No [] Yes []
EF2-11	Three or more Port Receive & Transmit EF2 Devices	B3.2.1	Devices having three or more ports have two ports wired as a loop through with an attached responder transmitter	M	No [] Yes []
EF2-12	EF2 Bi-directional Distribution Amplifiers	B3.2.2	Return data receivers "wire-OR" connected within the unit	M	No [] Yes [] N/A[]
EF2-13	EF2 Bi-directional Distribution Amplifiers	B3.2.2	No processing of data performed by a bi-directional distribution amplifier	M	No [] Yes []
EF2-14	EF2 Data Link termination	B3.2.3 / 4.9	Termination of EF2 primary data links conforms to clause 4.9	M	No [] Yes []
EF2-15	EF2 Data Link termination	B3.2.3 / 4.9	EF2 systems terminate both ends of the secondary data link, conforming to 4.9	M	No [] Yes []
EF2-16	EF2 Data Link termination	B3.2.3	EF2 controllers and other transmitting devices provide a means to terminate the secondary data link	M	No [] Yes []
EF3	EF3	B3.3	EF3 systems use the primary data link to send a standard unidirectional DMX512 signal	M	No [] Yes [] N/A[]
EF3-1	EF3 Bi-directional Distribution Amplifiers	B3.3.1	Distribution Amp/Data Combiner control the direction of data flow on the secondary data link	M	No [] Yes []
EF3-2	EF3 Data Link termination	B3.3.2 / 4.9	Termination of EF3 primary data links conforms to clause 4.9	M	No [] Yes []
EF3-3	EF3 Data Link termination	B3.3.2 / 4.9	EF3 systems terminate both ends of the secondary data link, conforming to 4.9	M	No [] Yes []

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
EF3-4	EF3 Data Link termination	B3.3.2	EF3 controllers and other transmitting devices provide a means to terminate the secondary data link	M	No [] Yes []
EF4-1	EF4	B3.4 / B3.1	Both data links comply with the requirements of clauses B3.1 (EF1) and fall within the scope of EIA-485-A	M	No [] Yes [] N/A []
EF4-2	EF4	B3.4 / B3.1	use of the secondary data link is optional	M	No [] Yes [] N/A []

F3.3 Annex C - Higher Protection Levels (DMX512-A Protected)

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
HPT	Higher Protection Levels - Transmitters	C / Table C1	Protection Limits per Table C1	O	No [] Yes [] N/A []
HPR1	Higher Protection Levels - Receivers	C / Table C2	Protection Limits per Table C2	O	No [] Yes [] N/A []
HPR2	Higher Protection Levels - Receivers	C / Table C2 Note	Any resistance fitted between chassis and Data Link Common survives continuous connection to voltages within the EIA-485 Common Mode range of -7/+12 VDC	O	No [] Yes [] N/A []

F3.4 Annex D - Reserved Alternate START Codes

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
RASC1	Reserved Alternate START Codes	D1 / Table D1	Equipment does not generate Alternate START Codes 92 - A9 (146 - 169 decimal) or AB - CD (171 - 205 decimal)	M	No [] Yes [] N/A []
RASC2	Reserved Alternate START Codes - Development	D1 / Table D1	No advertisement or sale of devices that use Alternate START Codes F0 - F7 (240 - 247 decimal)	M	No [] Yes []
RASC3	Text packet	D2	ASC 17h (23 decimal) used for text packet	O	No [] Yes [] N/A []

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
RASC4	Text packet	D2	Packet implements ISO/IEC 646	M	No [] Yes []
RASC5	Text packet	D2	Slot value of zero terminates the ASCII string	M	No [] Yes []
RASC6	Text packet	D2	Slots transmitted after this null terminator up to the reset sequence are ignored	M	No [] Yes []
RASC7	Test packet format	D3	ASC 55h (85 decimal) designates special test packet of 512 data slots carrying the value 55h (85 decimal)	O	No [] Yes [] N/A []
RASC8	Test packet time	D3	Test packet the time from the start of the Break until the stop bit of the 513th slot is no more than 25 milliseconds	M	No [] Yes [] Value if Yes: _____
RASC9	Test packet Back-to-Back	D3	When test packets are sent back to back, MBB time no more than 88 μ sec	M	No [] Yes [] Value if Yes: _____
RASC10	Test packet Break time	D3	Break timing for test packets shall be $\geq 88 \mu\text{sec} \leq 120 \mu\text{sec}$	M	No [] Yes [] Value if Yes: _____
RASC11	Test packet MAB	D3	The MAB time $\geq 8 \mu\text{sec} \leq 16 \mu\text{sec}$		No [] Yes [] Value if Yes: _____
SIP	System Information Packet (SIP)	D4	No other packet sent between the NSC packet and the SIP (ASC CFh - 207 decimal) that carries its checksum	O	No [] Yes [] N/A []
SIP01	System Information Packet (SIP) format	D4.2 / Table D2	Receivers accepts SIPs of up to 255 data slots	M	No [] Yes []
SIP02	System Information Packet (SIP) - Transmitter	D4.3	Transmitting devices sends a value of 24 in slot 1	M	No [] Yes []
SIP03	System Information Packet (SIP) - Receiver	D4.3	Receivers use the value received in slot 1 to establish the offset to the SIP checksum	M	No [] Yes []

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
SIP04	System Information Packet (SIP)	D4 / D4.4.1	If implemented, when the subsequent NULL packet hold Control Bit is set (=1), the subsequent NULL START Code packet held pending the reception of the next SIP for validation of the checksum	M	No [] Yes [] N/A[]
SIP05	System Information Packet (SIP)	D4 / D4.4.1	If a second NULL START Code packet is received without a preceding SIP, the receiver returns to an immediate use mode	M	No [] Yes []
SIP06	System Information Packet (SIP) - DMX512 Processing Level	D4.8 / Table D2	Originating devices transmits a value of 00h in slot 7	M	No [] Yes []
SIP07	System Information Packet (SIP) - DMX512 Processing Level	D4.8 / Table D2	Processing devices increment the value of this field by 01h	M	No [] Yes [] N/A[] N/A because: _____
SIP08	System Information Packet (SIP) - Packet History	D4.13	DMX512 Processing devices and media converters insert their own Manufacturer's ID into the SIP packet	M	No [] Yes []
SIP09	System Information Packet (SIP) - Packet History	D4.13	Originating device sends its Manufacturer's ID in SIP slots 13 and 14, with 0000h in slots 15, 16; 17, 18; 19, 20 and 21, 22	M	No [] Yes [] N/A[]
SIP10	System Information Packet (SIP) - Packet History	D4.13	Subsequent processing devices insert their manufacturer's ID into the slots as indicated by the DMX512 processing level slot	M	No [] Yes [] N/A[]

F3.5 Annex E - Alternate START Code, Manufacturer ID, and Enhanced Functionality Registration

Item	Feature	Clause	Value / Comment	Mandatory / Optional	Supported
ASC	Alternate START Codes	E	ASC not covered by Annex D are used	O	No [] Yes [] N/A []
AC1	ASC Registration	E4	ASC Registered through E1 Secretariat	M	No [] Yes [] N/A []
AC2	ASC Registration	E6.1	2-line description of the purpose of the Alternate START Code provided	M	No [] Yes [] N/A []
AC3	ASC Registration	E6.1	Minimum and maximum number of slots, including the START Code, in any proposed packet listed	M	No [] Yes [] N/A []
AC4	ASC Register - by E1 Secretariat	E6.1	Description (subject to editing) has been provided to the E1 Secretariat for inclusion in the Register	M	No [] Yes [] N/A []
AC5	ASC Register - by E1 Secretariat	E6.2	E1 Secretariat maintains Register	M	No [] Yes [] N/A []

Contact Information

E1 Secretariat

Entertainment Services and Technology Association

Karl G. Ruling

Technical Standards Manager

ESTA

875 Sixth Avenue, Suite 1005

New York, NY 10001

Phone: +1-212-244-1505

FAX: +1-212-244-1502

email: kruling@esta.org

Technical Standards Committee Chairperson

Mr. Mike Garl

President

James Thomas Engineering, Inc.

10240 Caneel Drive

Knoxville, TN 37931

Phone: +1-865-692-3060

FAX: +1-865-692-9020

email: mikeg@jthomaseng.com

Control Protocols Working Group Chairpersons

Steve Carlson (thru 2003)

High Speed Design, Inc.

11929 N. W. Old Quarry Road

Portland, OR 97229

Phone: +1-503-626-4206

FAX: +1-503-626-4206

email: scarlson@hspdesign.com

Tracy Underhill (2004)

Electronics Diversified Inc.

1099 S. Walden Way #205

Aurora, CO 80017

Phone: +1-303-306-9170

FAX: +1-303-306-7636

email: tracy.underhill@edionline.com

Steve Terry

Electronic Theatre Controls, Inc.

630 Ninth Avenue, Suite 1001

New York, NY 10036

Phone: +1-212-397-8080

FAX: +1-212-397-4340

email: sterry@etcconnect.com

Acknowledgments:

Control Protocols Working Group - DMX512 Task Group Members:

Chair/Editor: Mitch Hefter, Entertainment Technology; USITT (USA)

Tim Bachman, A.C.T. Lighting (USA)

Scott Blair (USA)

Javid Butler, Mainstage (USA)

Steve Carlson, High Speed Design (USA)

Tim Cox, PLASA (UK)

Milton Davis, Doug Fleenor Design (USA)

Tony Douglas-Beveridge, PLASA (UK)

Doug Fleenor, Doug Fleenor Design (USA)

Bob Goddard, Goddard Design (USA)

Dave Higgins, Pathway Connectivity (Canada)

Wayne Howell, Artistic Licence (UK)

Paul Mardon, Pulsar Light of Cambridge (UK)

Ted Paget, Jones & Phillips (USA)

Eckart Steffens, Soundlight; VPLT (Germany)

Steve Terry, Electronic Theatre Controls (USA)

Michael A. (Sandy) Twose, Sand Systems (Canada)

Steve Unwin, Pulsar Light of Cambridge (UK)

Peter Willis, Howard Eaton Lighting Ltd. (UK)

Control Protocols Working Group

Principal voting members at the time this document was approved by the Working Group:

Tim Bachman; A.C.T Lighting, Inc. [DR]
Arnold Tang; Arnold Tang Productions [U]
Wayne David Howell; Artistic Licence (UK) Ltd. [CP]
Tobin Neis; Barbizon Companies [DR]
George Kindler; Carter & Burgess, Inc. [G]
Doug Fleenor; Doug Fleenor Design, Inc. [MP]
Daniel W. Antonuk; Electronic Theatre Controls, Inc. [MP]
Tracy Underhill; Electronics Diversified Inc. [MP]
Alfred Langguth; Embedded Controls by Design, LLC [MP]
Philip Nye; Engineering Arts [G]
Robert Goddard; Goddard Design Co. [MP]
Scott M. Blair; High End Systems Inc. [MP]
Steve Carlson; High Speed Design, Inc. [MP]
Alan Martello; Horizon Control Inc. [MP]
Peter Willis; Howard Eaton Lighting Ltd. [CP]
Richard Thornton Brown; i Light Group PLC [MP]
Edwin S. Kramer; I.A.T.S.E. Local 1 [U]
Roger Lattin; I.A.T.S.E. Local 728 [U]
Rick Leinen; Leviton Manufacturing Co., Inc. [MP]
John (Javid) D. Butler; Mainstage Theatrical Supply [DR]
Ole Bystrup; Martin Professional A/S [MP]
Gary Douglas; Pathway Connectivity Inc. [MP]
James Eade; PLASA [G]
Hans Lau; Sand Network Systems [MP]
Michael Gonzales; Spectrum Lighting Inc. [DR]
Richard Lawrence; Strand Lighting Ltd. [MP]
Brian Dowd; TMB [U]
Mitch Hefter; USITT [U]
Charles Reese; VLPS [MP]
Mark Jensen; Wybron, Inc. [MP]

Alternate voting members:

Simon Hobday; Artistic Licence (UK) Ltd. [CP]
Ken Wagner; Doug Fleenor Design, Inc. [MP]
Milton Davis; Doug Fleenor Design, Inc. [MP]
Steve Terry; Electronic Theatre Controls, Inc. [MP]
James Johnson; Electronics Diversified Inc. [MP]
Tom Grimes; High End Systems, Inc. [MP]
Robert Bell; Horizon Control Inc. [MP]
John Huntington; I.A.T.S.E. Local 1 [U]
Alan M. Rowe; I.A.T.S.E. Local 728 [U]
Dennis Grow; I.A.T.S.E. Local 728 [U]
Ken Vannice; Leviton Manufacturing Co., Inc. [MP]
Dave Higgins; Pathway Connectivity Inc. [MP]
Kevin Loewen; Pathway Connectivity Inc. [MP]
Tony Douglas-Beveridge; PLASA [G]
Michael A. (Sandy) Twose; Sand Network Systems [MP]
Yngve Sandboe; Sand Network Systems [MP]
Michael Lay; Strand Lighting Ltd. [MP]

Observer members:

Andre Broucke; ADB-TTV Group [MP]
John Sellers; AIM Northwest [G]
Steve Friedlander; Auerbach Pollock Friedlander [U]
Raymond Wynn; AVL Systems Pty Ltd. [MP]
J. B. Toby; Avolites Ltd. [MP]
Richard Salzedo; Avolites Ltd. [MP]
Shahid Anwar; Avolites Ltd. [MP]
Barbara Wohlsen; Barbara Wohlsen [U]
Bernardo Benito Rico; Ben-Ri Electronica S.A. [MP]
Lee J. Bloch; Blue Sky Design, Ltd. [G]
Soo-Myong Chung; Blue Sky Design, Ltd. [G]
Bradley Klinkradt; Bradley Klinkradt [G]
Ted Fregon; Bytecraft Pty. Ltd. [MP]
Bill Ellis; Candela Controls, Inc. [U]
Rick Szijarto; Cast Group of Companies, Inc. [CP]
Marty Lazarus; Chicago Spotlight, Inc. [G]
Larry Dunn; City Theatrical, Inc. [CP]
Fabiano Pina; Clay Paky S.P.A. [MP]
Woody Smith; Coemar USA Inc. [MP]
Ohad Ashery; Compulite Systems [MP]
Simeon Aladjem; Compulite Systems [MP]
Yehuda Shukram; Compulite Systems [MP]
Dietmar Rottinghaus; Connex GmbH [G]
Jason Friedman; Creative Realities, Inc. [G]
David Bertenshaw; David Bertenshaw [G]
Bob Toms; Design Advantage LLC [G]
Gary Dove; Dove Systems [MP]
Jussi Kallioinen; Eastway Sound & Lighting [U]
Ed Jones; Edwin Jones Co., Inc. [CP]
Joost van Eenbergen; ELC Lighting [MP]
Bill Fehrmann; Electrol Engineering, Inc. [MP]
Adam Bennette; Electronic Theatre Controls, Inc. [MP]
Anders Ekvall; Electronic Theatre Controls, Inc. [MP]
Hans Leiter; Electronic Theatre Controls, Inc. [MP]
Erwin Rol; Erwin Rol [G]
Sierk Janszen; Ground Zero [U]
Liao Wei Min; GuangZhou HeDong Electronic Co., Ltd. [G]
Alan P. Symonds; Harvard University [U]
Trevor Forrest; Helvar Lighting Control [MP]
Bill Hewlett; Hewlett Electronics [CP]
Gian Carlo C. Bartolotti; Ibeam SP / Banco de Eventos [U]
Geoffrey O. Thompson; IEEE 802.3/Nortel Networks [G]
Rob Johnston; Interactive Technologies, Inc. [MP]
Larry Schoeneman; Interesting Products, Inc. [G]
David Timmins; Jands Electronics [MP]
Helge Hoffmann; JB Lighting [MP]
Christopher Purpura; Jones & Phillips Associates, Inc. [G]
Edward Paget; Jones & Phillips Associates, Inc. [G]
John Mehlretter; Lehigh Electric Products Co. [MP]
Mark T. Kraft; Lehigh Electric Products Co. [MP]
William L. Maiman; Lex Products Corp. [MP]
Klaus Amling; Licht-Technik [P]
Andrew Sherar; Lightmoves PLC [MP]
Simon Alpert; Lighttech Event Technologies [CP]
Gary Pritchard; LSC Lighting Systems PTY Ltd [MP]
Bart Swinnen; Luminex LCE [MP]
J. P. Steiner; Lutron Electronics [MP]
Jim Holladay; Luxence [G]
Hiroshi Kita; Marumo Electric Co., Ltd. [MP]
Petri Laine; Obelux Oy [CP]
Edward R. Condit; On Location Lighting Systems [DR]
Stuart Cotts; Oregon Shakespeare Festival [U]
David A. Boller; Organic Machines LLC [CP]
William Benner; Pangolin Laser Systems [MP]
Mac Perkins; PNTA Inc. [G]
Paul F. Mardon; Pulsar Ltd. [MP]
Steve Unwin; Pulsar Ltd. [MP]
Stephen J. Tyrrell; Quantum Logic [MP]
Douglas Franz; QVC Network [U]
Charlie Richmond; Richmond Sound Design Ltd. [CP]
Martin Farnik; Robe Show Lighting s.r.o. [MP]
Kenneth Mockler; Scéno Plus [U]
Mick Martin; ShowCAD Control Systems [MP]
Loren Wilton; Showman Systems [CP]
Jon R. Farley; Sixteenth Avenue Systems [CP]
Robert Barbagallo; Solotech Inc. [U]
Ujjal Kar; Standard Robotics & Lighting [G]
Paul K. Ericson; Syska Hennessy Group Lighting Design [U]
Stephen Bickford; T. Kondos Associates [U]
Tad Trylski; Tad Trylski [U]
Klas Dalbjorn; TC Group [MP]
Jerry Gorrell; Theatre Safety Programs [G]
Colin Waters; TMB [U]
Jason Potterf; University of Texas at Austin [G]
Matt Stroud; University of Texas at Austin [G]
Torrey Bievenour; Vision Quest Lighting [G]
Eckart Steffens; VPLT [G]
Eric Cornwell; West Side Systems [U]
John Sondericker III; Wybron, Inc. [MP]

[CP] = custom-market producer [DR] = dealer rental company [G] = general interest
[MP] = mass-market producer [U] = user



worldwide standards for the entertainment industries

American National Standard
E1.11 – 2008 (R2013)
Entertainment Technology
USITT DMX512-A
Asynchronous Serial Digital Data
Transmission Standard for Controlling
Lighting Equipment and Accessories

CP/2007-1013r3.1

This document was approved as an American National Standard by the ANSI Board of Standards Review on 13 March 2013. It is a reaffirmation of the 2008 edition of the Standard.

© 2013 PLASA North America. All rights reserved.

Notice and Disclaimer

PLASA does not approve, inspect, or certify any installations, procedures, equipment or materials for compliance with codes, recommended practices or standards. Compliance with a PLASA standard or recommended practice, or an American National Standard developed by PLASA, 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 PLASA.

PLASA 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, PLASA 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:

PLASA North America
630 Ninth Avenue, Suite 609
New York, NY 10036 USA
Phone: 1-212-244-1505
Fax: 1-212-244-1502
standards.na@plasa.org
<http://tsp.plasa.org/>

For additional copies of this document contact:

The ESTA Foundation
630 Ninth Avenue, Suite 609
New York, NY 10036 USA
Phone: 1-212-244-1421
Fax: 1-212-244-1502
<http://www.estafoundation.org>

The PLASA Technical Standards Program

The PLASA Technical Standards Program was created to serve the PLASA 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. PLASA works closely with the technical standards efforts of other organizations within our industry, including USITT, PLASA, and VPLT, as well as representing the interests of PLASA 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: Control Protocols, Electrical Power, Floors, Fog and Smoke, Followspot Position, Photometrics, Rigging, and Stage Lifts.

PLASA 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 two hundred people, you must complete an application which is available from the PLASA 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 PLASA 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. PLASA is committed to developing consensus-based standards and recommended practices in an open setting.

Contact Information

Technical Standards Manager

Karl G. Ruling
PLASA North America
630 Ninth Avenue, Suite 609
New York, NY 10036
USA
1-212-244-1505
karl.ruling@plasa.org

Technical Standards Council Chairpersons

Mike Garl
President and General Manager
Tomcat USA
2160 Commerce Drive
PO Box 550
Midland, TX 79703-7504
USA
1-432-694-7070
mike.garl@tomcatusa.com

Mike Wood
Mike Wood Consulting LLC
6401 Clairmont Drive
Austin, TX 78749
USA
1-512-288-4916
mike@mikewoodconsulting.com

Control Protocols Working Group Chairpersons

Michael Lay
Philips Color Kinetics
3 Burlington Woods Drive
Burlington, MA 01803
USA
Phone: 1-781-418-9145
michael.lay@philips.com

Kimberly Corbett
Schuler Shook.
325 North Saint Paul, Suite 3250
Dallas, TX 75201
USA
Phone: 1-214-747-8300
kcorbett@schulershook.com

Acknowledgments

The Control Protocols Working Group was the consensus body for the development of this Standard. The working group's membership at the time the working group approved this Standard on 7 January 2013 is listed below.

Voting members:

Daniel W. Antonuk; Electronic Theatre Controls, Inc.; MP
Robert Bell; Acuity Brands Inc.; MP
Marcus Bengtsson; LumenRadio AB; MP
Scott M. Blair; Full Throttle Films/ VER; DR
Ron Bonner; PLASA EU; G
Jean-Francois Canuel; A.C. Lighting Ltd.; CP
Kimberly Corbett; Schuler Shook; DE
Milton Davis; Doug Fleenor Design, Inc.; MP
Gary Douglas; Acuity Brands Inc.; MP
Hamish Dumbreck; James Embedded Systems Engineering; MP
Larry Dunn; City Theatrical, Inc.; MP
Doug Fleenor; Doug Fleenor Design, Inc.; MP
Howard Forryan; Harting KGAA; G
Garrett Gallant; Acuity Brands Inc.; MP
Ed Garstkiewicz; Harting KGAA; G
Robert Goddard; Goddard Design Co.; MP
Dennis Grow; I.A.T.S.E. Local 728; U
Mitch Hefter; USITT; U
Simon Hobday; Artistic Licence Holdings; DE
Wayne David Howell; Artistic Licence Holdings; DE
John Huntington; I.A.T.S.E. Local 1; U
Michael Karlsson; LumenRadio AB; MP
Reza Khanmalek; A.C. Lighting Ltd.; CP
Insu Kim; Electronics and Telecommunications Research Institute; G
Peter Kirkup; Lumen Radio AB; MP
Paul Kleissler; City Theatrical, Inc.; MP
Edwin S. Kramer; I.A.T.S.E. Local 1; U
Ulrich Kunkel; DIN, NVBF Committee; U
Roger Lattin; I.A.T.S.E. Local 728; U
Hans Lau; LumenRadio AB; MP
Michael Lay; Royal Philips Electronics; MP
Sang-Kyu Lim; Electronics and Telecommunications Research Institute; G
Joshua Liposky; Lex Products Corp.; CP
Kevin Loewen; Acuity Brands Inc.; MP
Alan Martello; Acuity Brands Inc.; MP
Tyrone Mellon Jr.; Lex Products Corp.; CP
Simon Newton; Simon Newton; G
Maya Nigrosh; Electronic Theatre Controls, Inc.; MP
Philip Nye; Acuity Brands Inc.; MP
Edward A. (Ted) Paget; Daktronics Inc.; G
Charles Reese; Production Resource Group; DR
Alan M. Rowe; I.A.T.S.E. Local 728; U
Larry Schoeneman; DesignLab Chicago, Inc.; DR
Sean Sill; Doug Fleenor Design, Inc.; MP
Steve Terry; Electronic Theatre Controls, Inc.; MP
Christopher Tilton; Westlake Reed Leskosky; DE
Ken Vannice; Leviton Manufacturing Co., Inc.; MP
Peter Willis; Howard Eaton Lighting Ltd.; CP

Observer members:

Charles Alicea; Royal Philips Electronics; MP
Simon Alpert; Lighttech Event Technologies; CP
Klaus Amling; Licht-Technik; MP
Shahid Anwar; Avolites Ltd.; MP
Matthew Ardine; IATSE Local 728; U
Daniel Ayers; Norcostco; DR
Timothy Baarsch; Artistic Licence Holdings; DE
Robert Barbagallo; Solotech Inc.; U
Adam Bennette; Electronic Theatre Controls, Inc.; MP
David Bertenshaw; G
Stephen Bickford; T. Kondos Associates; U
Torrey Bievenour; Vision Quest Lighting; G
Lee J. Bloch; Bloch Design Group, Inc.; G
David A. Boller; Organic Machines LLC; CP
André Broucke; G
John (Javid) D. Butler; Integrated Theatre, Inc.; CP
Steve Carlson; High Speed Design, Inc.; MP
Nyasha Chigwamba; Rhodes University; U
Soo-Myong Chung; Bloch Design Group, Inc.; G
Paul J. Clark; HxDx; CP
Edward R. Condit; Edward R. Condit; G
Eric Cornwell; West Side Systems; U
Stuart Cotts; Oregon Shakespeare Festival; U
Klas Dalbjorn; TC Group; MP
Ben Darrington; Wireless Solutions Sweden AB; MP
Gilray Densham; CAST Group Inc; MP
Gary Dove; Dove Systems; MP
Yongshun Duan; Macostar International Ltd.; CP
Jerry Durand; Durand Interstellar, Inc.; CP
James Eade; PLASA; G
Andrew Eales; Rhodes University; U
Bill Ellis; Candela Controls, Inc.; U
Paul K. Ericson; Syska Hennessy Group Lighting Design; U
Jon R. Farley; Sixteenth Avenue Systems; CP
Martin Farnik; Robe Show Lighting s.r.o.; MP
Trevor Forrest; Helvar Lighting Control; MP
Steve Friedlander; Auerbach Pollock Friedlander; U
Phillip M. Gallo; TMB; DR
Jerry Gorrell; Theatre Safety Programs; G
Tom Grimes; Barco; MP
Josh Gubler; CP
Rob Halliday; U
Sean Harding; High Output, Inc.; G
Bill Hewlett; Hewlett Electronics; CP
Helge Hoffmann; JB Lighting; MP
Jim Holladay; Luxence; G
Osedum P. Igumbor; Rhodes University; U
Sierk Janszen; Ground Zero; U
Eric Johnson; G
Rob Johnston; Interactive Technologies, Inc.; MP
Ed Jones; Edwin Jones Co., Inc.; CP
Jussi Kallioinen; Eastway Sound & Lighting; U
Tae Gyu Kang; Electronics and Telecommunications Research Institute; G
Hiroshi Kita; Marumo Electric Co., Ltd.; MP
Phil Klapwyk; IATSE Local 891; U

Mark T. Kraft; Lehigh Electric Products Co.; MP
 Jason Kyle; JPK Systems Ltd.; MP
 Rick Leinen; Leviton Manufacturing Co., Inc.; MP
 Hans Leiter; Electronic Theatre Controls, Inc.; MP
 Jon Lenard; Applied Electronics; MP
 Maarten Lepelaars; eldoLED; MP
 Mark Manthei; Shure Inc.; G
 Paul F. Mardon; Pulsar Ltd.; MP
 Mick Martin; ShowCAD Control Systems; MP
 Paul Kenneth McEwan; Cooper Controls Ltd.; MP
 John Mehlretter; Lehigh Electric Products Co.; MP
 Avraham Mendall Mor "Avi"; Lightswitch; U
 John Musarra; U
 Tobin Neis; Barbizon Companies; DR
 Lars F. Paape; Scientific Algorithms and Embedded Systems; U
 Gary Pritchard; LSC Lighting Systems PTY Ltd; MP
 Eric Proce; U
 Torben Kaas Rasmussen; Martin Professional A/S; G
 Charlie Richmond; Richmond Sound Design Ltd.; CP
 Bernardo Benito Rico; Ben-Ri Electronica S.A.; MP
 Steve Roberts; Carr & Angier; G
 Erwin Rol; G
 Dietmar Rottinghaus; Connex GmbH; G
 Richard Salzedo; Avolites Ltd.; MP
 Yngve Sandboe; Sand Network Systems, Inc.; MP
 Martin Searancke; Dream Solutions Ltd.; MP
 Chuck Seifried; City of Phoenix; U
 John Sellers; AIM Northwest; G
 Andrew Sherar; Lightmoves PLC; MP
 Yehuda Shukram; Compulite Systems; MP
 Storm K. Staley; Stormwerx; U
 Ralph Stillinger; Royal Philips Electronics; MP
 Bart Swinnen; Luminex LCE; MP
 Arnold Tang; Arnold Tang Productions; U
 Geoffrey O. Thompson; IEEE 802.3/Nortel Networks; G
 David Timmins; Jands Electronics; MP
 J. B. Toby; Avolites Ltd.; MP
 Bob Toms; Catalyst Microsystems LLC; G
 Robert Tooker; U
 Tad Trylski; Tad Trylski; U
 Stephen J. Tyrrell; Quantum Logic; MP
 Tracy Underhill; 4U Consulting; G
 Steve Unwin; Pulsar Ltd.; MP
 Samuli Valo; Picturall Ltd.; MP
 Carlo Venturati; Clay Paky S.P.A.; MP
 Will Wagner; Carallon Ltd.; MP
 Oliver Waits; Avolites Ltd.; MP
 John Warwick; Royal Philips Electronics; MP
 Colin Waters; TMB; DR
 Ralph Weber; ENDL Texas; G
 Lars Wernlund; Lewlight; MP
 Michael (Mike) Whetstone; Integrated Theatre, Inc.; CP
 Loren Wilton; Showman Systems; CP
 Barbara Wohlsen; Barbara Wohlsen; U
 C. S. Wong; Macostar International Ltd.; CP
 Jiantong Wu; Beijing Special Engineering Design & Research Institute; G

Kehang Wu; Shure Inc.; G
David Yellin; LightMinded Industries, Inc.; MP
Larry Zoll; Zoll Design & Consulting, LLC; U

CP	Custom-market Producer
DE	DEsigner incorporating the subject matter of the working group in projects
DR	Dealer or Rental company
G	General interest
MP	Mass-market Producer
U	User

Contents

Notice and Disclaimer	i
Contact Information.....	iii
Acknowledgments.....	iv
Contents	viii
Foreword	xii
1 General.....	1
1.1 Scope.....	1
1.2 Overview and Architecture	1
1.3 Appropriate uses of this Standard	1
1.4 Classes of data appropriate for transmission over links designed to this Standard	2
1.5 Classes of data not appropriate for transmission over links designed to this Standard	2
1.6 Compliance	2
2 Normative references.....	2
3 Definitions.....	3
4 Electrical Specifications and Physical Layer	5
4.1 General	5
4.2 Electrical isolation.....	5
4.3 Topology	5
4.4 DMX512 ports.....	5
4.5 Data link common and grounding topologies	6
4.6 Preferred method of earth grounding data link common	6
4.7 Primary data link.....	6
4.8 Secondary data link	6
4.8.1 Secondary data link - active use	6
4.8.2 Secondary data link - passive loop through ports.....	6
4.9 Data link termination procedures.....	6
4.10 Unpowered devices	6
5 Nominal Operating Characteristics	7
5.1 General	7
5.2 Chassis in power isolated equipment	7
5.3 Earth grounding of data link common for transmitters	7
5.4 Ground referenced transmitters.....	7
5.5 Disallowed transmitter topology.....	8
5.6 Earth grounding of data link common for receivers	9
5.8 Disallowed receiver topology.....	10
5.9 DMX512 processing devices.....	11
5.10 Loading designation.....	11

6 Protection	11
6.1 Minimum protection against interconnection damage	11
6.2 Minimum Electro Static Discharge (ESD) protection	12
7 Connection Methods	12
7.1 Equipment fitted with user accessible pluggable data link connections	12
7.1.1 Required connector	12
7.1.2 Concession for use of an alternate connector (NCC DMX512-A)	12
7.2 Equipment intended for fixed installation with internal connections to the data link.....	12
7.3 IEC 60603-7 8-position modular connectors.....	13
8 Data protocol	13
8.1 Format.....	13
8.2 Slot format.....	13
8.3 Break.....	14
8.4 Mark after break	14
8.5 START code.....	14
8.5.1 NULL START code.....	14
8.5.2 Dimmer class data.....	14
8.5.3 Other START codes	15
8.5.4 START code processing.....	15
8.6 Maximum number of data slots	15
8.7 Minimum number of data slots	16
8.8 Defined line state between slots.....	16
8.9 Defined line state between data packets (Mark Before Break)	16
8.10 Break-to-Break spacing.....	16
8.11 Timing Diagram - data+	16
9 Receiver Performance.....	18
9.1 Rejection of Improperly framed slots	18
9.2 Loss of data tolerance / Resumption of acceptance of data	18
9.3 Receiver performance at maximum refresh rate	18
9.4 Packet processing latency	18
10 Marking and Disclosures	19
10.1 Identification	19
10.2 DMX512 port marking.....	19
10.3 Data line termination marking.....	19
10.4 Ground / Isolation marking	19
10.5 Required disclosures and markings.....	20
10.5.1 Portable products and products fitted with external pluggable data link connectors	20
10.5.2 Equipment intended for fixed installation with internal connections to the data link	20

10.5.3 Loss of data handling procedure.....	20
10.5.4 Packet processing latency	20
10.5.5 NULL START Code functionality.....	20
10.5.6 Slot footprint.....	20
Annex A - Non Preferred (Alternate) topologies (Normative).....	21
A1 Isolated transmitters	21
A2 Non-isolated receivers.....	22
A3 Grounded Receivers	23
A4 Earth grounding of data link common for floating devices.....	24
Annex B (Normative) - Enhanced DMX512.....	26
B1 General.....	26
B2 Summary of Enhanced Function Topologies.....	26
B3 Identification of data protocols for Enhanced DMX512	26
B3.1 EF1 Half Duplex DMX512 - Bidirectional use of the primary data link	27
B3.2 EF2 - Full duplex DMX512	27
B3.3 EF3 Half duplex on the second data link	28
B3.4 EF4 Protocols that use the primary and the secondary data links in ways not covered above ..	28
B3.5 Additional electrical requirements.....	28
Annex C (Normative) - Higher Protection Levels – “DMX512-A Protected”.....	29
Annex D (Normative) - Reserved Alternate START Codes	30
D1 Reserved Alternate START Codes	30
D2 ASC text packet	30
D3 ASC test packet	31
D4 UTF-8 text packet.....	31
D5 System Information Packet (SIP) Alternate START Code	31
D5.1 Application.....	31
D5.2 SIP format	31
D5.3 SIP checksum pointer.....	31
D5.4 Control bit field.....	32
D5.5 Checksums.....	33
D5.6 SIP Sequence number.....	33
D5.7 Originating universe.....	33
D5.8 DMX512 processing level	33
D5.9 Software version.....	33
D5.10 Packet lengths.....	33
D5.11 Number of packets.....	33
D5.12 Manufacturer ID.....	33
D5.13 Packet history.....	33

D5.14 SIP Checksum	34
Annex E (Normative) - Alternate START Code, Manufacturer ID, and Enhanced Functionality Registration	35
E1 Alternate START Code Registration Policy: 1 - 255 decimal (01 - FF hexadecimal).....	35
E2 Authorized use	35
E3 Reserved Alternate START Codes	35
E4 Requests for Registration of New START Codes	35
E4.1 Number of Alternate START Codes per entity	35
E4.2 Selection of the Alternate START Code value and Manufacturer ID	35
E5 Requirements for registration of an EF protocol	35
E6 Documentation Register	35
E6.1 Documentation for use of Alternate START Codes.....	35
E6.2 Maintenance and Publication	35
E6.3 Supplemental documentation.....	36
E7 Ownership	36

Foreword

(This foreword contains no requirements and is not part of E1.11.)

This Standard describes a method of digital data transmission between controllers and controlled lighting equipment and accessories, including dimmers and related equipment. This Standard is intended to provide for interoperability at both communication and mechanical levels with controllers made by different manufacturers.

There are five normative annexes in this Standard. These address extensions of the base standard and are considered part of the Standard, which means that when an extension described in an Annex is implemented, compliance with the annex is mandatory. However, a product compliant with the Standard can be manufactured without implementing these annexes.

The original version of the DMX512 Standard was developed in 1986 by the Engineering Commission of the United States Institute for Theatre Technology, Inc. (USITT). Minor revisions were made in 1990. DMX512 has gained international acceptance throughout the entertainment industry, even though USITT is not formally accredited as a standards making body. The earlier versions of this Standard covered only data used by dimmers. In practice this Standard has been used by a wide variety of devices; this version recognizes this fact.

In 1998, it became evident that additional updates to the Standard were necessary and formal recognition through an internationally recognized standards organization was required. USITT issued a Call for Comments in order to solicit recommendations for changes to the Standard. At the same time, USITT transferred maintenance of DMX512 to ESTA's ANSI-accredited Technical Standards Program, now operating as PLASA's Technical Standards Program.

A Task Group established under the TSP's Control Protocols Working Group acted on the proposals received in response to the Call for Comments. The primary goal was to make editorial updates to DMX512 appropriate for current times, including the addition of technical features while maintaining a balance with backward compatibility. Many proposals, while technically innovative, could not be accepted because their implementation would not have been backward compatible and would have immediately rendered obsolete most of the installed base of equipment.

In 2004, as a result of the actions taken on those proposals and subsequent development under the *Policies and Procedures* of the ESTA Technical Standards Program, E1.11-2004 was approved as an American National Standard. Despite being an American National Standard, development has had strong international participation and support.

Beginning in 2007, based on comments and requests from users and manufacturers, minor revisions were made to E1.11-2004 and submitted to the public for review and comment. These changes included defining one of the reserved Alternate START Codes for UTF-8 text packets, a note emphasizing refresh timings, and removal of the informative PICS (Protocol Implementation Compliance Statement) clauses. The PICS clauses were simply a summary of the mandatory requirements of the standard, but often had to employ language that used double negatives and led to confusion about some requirements. The 2008 edition was the result of those revisions to the 2004 edition. This edition is a reaffirmation of the 2008 edition.

1 General

1.1 Scope

This Standard describes a method of digital data transmission between controllers and controlled equipment as described in Clause 1.4 and accessories, including dimmers. It covers electrical characteristics, data format, data protocol, and connector types.

This Standard is intended as a guide for:

1. Equipment manufacturers and system specifiers who wish to integrate systems of lighting equipment and accessories, including dimmers, with controllers made by different manufacturers.
2. Equipment manufacturers seeking to implement a standard digital transmission protocol in their lighting control and accessory products.
3. System specifiers and designers to gain detailed information about allowed connectors and allowed system topologies.

This standard is not intended to replace existing protocols other than USITT DMX512 and DMX512/1990. Cable requirements and premises wiring are not within the scope of this Standard.

Equipment compliant with this Standard will be marked DMX512-A or USITT DMX512-A in order to distinguish it from the previous (informally recognized) versions. Unless otherwise noted, references to DMX512 in this document refer to DMX512-A.

1.2 Overview and Architecture

This standard uses a simple asynchronous eight-bit serial protocol consisting of an untyped byte stream produced by standard UARTs. The physical media, not addressed in this document, is normally, but not exclusively, a two-pair cable, with each pair serving as a data link. The media is driven using ANSI/TIA/EIA-485-A-1998 (hereafter referred to as EIA-485-A in this document) balanced data transmission techniques. Physical connection at devices is via 5-pin XLR connectors or by “hard-wiring” to terminals. Restricted use of connectors other than 5-pin XLR is allowed if certain conditions apply (see clause 7).

Data on the primary data link is sent in packets of up to 513 slots. The first slot is a START Code, which defines the information in the subsequent slots in the packet. The interoperability of equipment complying with the Standard is largely due to the use of the NULL START Code by transmitting devices. Proper function is dependent upon the receiving device(s) extracting the pertinent data for processing from each transmitted packet.

Data on the secondary data link, when implemented, is used for a variety of purposes, all of which fall within the scope of EIA-485-A. Identification of the required circuit topology for any particular implementation is defined.

1.3 Appropriate uses of this Standard

Equipment designers and general users of this Standard will recognize that this Standard is intended to fill only a limited range of uses. Other standards will be more appropriate for different uses. This is not intended to support a venue wide network that can carry data for lighting, sound, and scenery mechanization, for example, all on the same wire.

This Standard does not require mandatory error checking of NULL START Code packets. There is no assurance that all DMX512 packets will be delivered. It is common practice for merge units and protocol converters to drop packets that they cannot process in a timely manner. The 1986 and 1990 versions of the USITT Standard specifically allow dimmers to ignore packets that they cannot process in a timely manner, and this concept survives in this version of the Standard with respect to NULL START Code packets.

1.4 Classes of data appropriate for transmission over links designed to this Standard

DMX512 is designed to carry repetitive control data from a single controller to one or more receivers. This protocol is intended to be used to control dimmers, other lighting devices and related non-hazardous effects equipment.

1.5 Classes of data not appropriate for transmission over links designed to this Standard

Since this Standard does not mandate error checking, DMX512 is not an appropriate control protocol for hazardous applications.

1.6 Compliance

Compliance with this Standard is strictly voluntary and the responsibility of the manufacturer. Markings and identification or other claims of compliance do not constitute certification or approval by the E1 Accredited Standards Committee. See clause 10 for Marking and Disclosure requirements.

2 Normative references

ANSI/TIA/EIA-568-B-2001	Commercial Building Telecommunications Cabling Standard
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
-------------	--

IEC 60603-7	Connectors for Frequencies Below 3 MHz for Use with Printed Wiring Boards - Part 7: Detail Specification for Connectors, 8-Way, Including Fixed and Free Connectors with Common Mating Features, with Assessed Quality
-------------	--

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

USITT DMX512/1990	Digital Data Transmission Standard for Dimmers and Controllers
-------------------	--

USITT
315 South Crouse Avenue – Suite 200
Syracuse, NY 13210
+1-800-938-7488 +1-315-463-6463
<http://www.usitt.org>

Fax: +1-315-463-6525

3 Definitions

3.1 Asynchronous: signals that start at any time and are not locked or synchronized to the receiving device by a separate clock line.

3.2 Balanced Transmission Line: a data communications line where two wires are present, the signal and its opposite (complement), the actual signal being the difference between the voltages on the two wires. Balanced lines have excellent noise and interference rejection properties.

3.3 Break: a high (mark) to low (space) transition followed by a low of at least 88 microseconds followed by a low to high transition.

3.4 Circuit Common: the common reference (zero volt supply) of the EIA-485-A driver or receiver circuitry.

3.5 Common: see Data Link, Signal Common, and Circuit Common.

3.6 Common Mode Voltage: a voltage appearing equally on the data+ (plus) and data- (minus) lines relative to circuit Common. $V_{cm} = (V_a + V_b)/2$ where:

V_{cm} is the Common Mode Voltage

V_a is the voltage on DMX512 data+ with respect to circuit Common

V_b is the voltage on DMX512 data- with respect to circuit Common

3.7 Controller: a transmitting device that originates DMX512 data.

3.8 Data+: signal true.

3.9 Data-: signal complement.

3.10 Data Link: physical connection between transmitting and receiving devices.

3.11 Data Link Common: the connection to circuit Common at the point of interconnection (DMX512 Port) of the product.

3.12 DMX512 Port: a DMX512 signal connection point (connector or terminal strip).

3.13 DMX512 Processing Device: a piece of equipment that regenerates the timing of any DMX512 packet or has provision for other signal inputs from which the outgoing DMX512 packet is generated. In the absence of any DMX512 transmitting capability, the device has provision for other signal outputs that are controlled in some manner by the incoming DMX512 packet. Basic buffer products are not normally considered processing devices.

3.14 Earth Ground: the common, zero potential available from the mains electricity supply and usually connected to the metal chassis of equipment. Earth Ground is referred to as Earth in Europe and Ground in the USA.

3.15 Enhanced Functionality: use of the optional secondary data link of a DMX512 port and/or optional additional use of the primary data link of a DMX512 port.

3.16 Idle: the time between slots that the DMX512 line is high and not sending any information (also known as the 'Mark' condition).

3.17 In-Line Device: any component that receives and re-transmits DMX512.

3.18 Isolated: circuit topology in which the output is completely electrically disconnected from the input.

3.19 Legacy (as used in this Standard): transmitting and receiving devices complying with the original USITT DMX512 or DMX512/1990 in all aspects of those standards. (Exception: receiving devices that are not dimmers but comply with all other aspects of DMX512/1990 are considered to be Legacy Equipment.)

3.20 Line Driver: an electrical circuit providing differential voltage excursions on a data link, operating within a defined Common Mode voltage range and with a specified response to overload and overvoltage conditions. Sometimes simply referred to as a “driver.”

3.21 Line Receiver: an electrical circuit allowing detection of differential voltage excursions on a data link, operating within a defined Common Mode voltage range and with a specified response to overload and overvoltage conditions.

3.22 Loop-Through Connection: a connector or terminal DMX512 port that connects the signals Data 1+, Data 1- and Data link common, and optionally Data 2+ and Data 2-, of one DMX512 port to another DMX512 port. Frequently abbreviated to Loop or Thru.

3.23 Manufacturer ID: a two-byte value assigned to a Manufacturer/Organization by the E1 Accredited Standards Committee for use with specific Alternate START Codes. This ID identifies the data contained in the packet as proprietary. This packet may be safely ignored by systems that do not implement the specific start code.

3.24 Mark: a line condition where Signal True is high with respect to Signal Complement. A Mark represents a binary 1.

3.25 MAB (MaB): Mark After Break – the period of time between the low to high transition that signifies the end of Break and the high to low transition which is the start bit of the START Code (slot 0).

3.26 MBB (MbB): Mark Before Break – the period of time between the end of the second stop bit of the last slot and the high to low transition that signifies the start of Break.

3.27 Merge Unit: a product comprising one or more receiving devices and one or more transmitting devices that generate a DMX512 packet derived from the manufacturers declared logical combination of the DMX512 input packets.

3.28 NULL START Code: a START Code with a value of zero (00h).

3.29 Packet (in DMX512-A): a Reset Sequence followed by all slots up to the Mark Before Break.

3.30 Receiver (Receiving Device): a piece of equipment that accepts a DMX512 signal.

3.31 Refresh Rate: the number of DMX512 packets with a NULL START Code sent per second.

3.32 Reset Sequence: a sequence of a Break, Mark After Break, and START Code.

3.33 Signal Common: the common reference conductor of the physical media (e.g., the cable shield).

3.34 Slot: a sequentially numbered framed byte in a DMX512 packet. A single Universe contains a maximum of 513 Slots, starting at slot 0. Slot 0 is the START Code. Slots 1 through 512 are data slots.

3.35 Slot Footprint: the number of data slots used by a product in its operation.

Note: A 24 way dimmer rack may have a footprint of 24, it may be more if some slots are used to provide additional control functions using NULL START Code packets. Automated luminaires usually require a Slot Footprint of greater than one.

3.36 Space: a line condition where Signal True is low with respect to Signal Complement. A Space represents a binary 0.

3.37 Start Bit: the additional bit attached to the beginning of a byte to indicate to the receiver that a new byte is being sent. The start bit is always low, i.e., Space.

3.38 START Code: the first slot sent after Break, indicating the type of information to follow.

3.39 Stop Bit: the additional bit(s) attached to a byte to indicate the end of the byte – DMX512 has 2 stop bits. The stop bit is always high, i.e., Mark.

3.40 Terminator: a device or circuit topology that is designed to minimize unwanted signal reflections on a data link.

3.41 Transmitting Device: a piece of equipment that produces a DMX512 signal.

3.42 UART: Universal Asynchronous Receiver/Transmitter. A device that generates and/or decodes serially transmitted data.

3.43 Universe: a DMX512 data link originating from a single DMX512 source. Control of up to 512 DMX512 data slots comprises a single universe.

4 Electrical Specifications and Physical Layer

4.1 General

The electrical specifications of this Standard are those of EIA-485-A, except where specifically stated in this document. Where a conflict between EIA-485-A and this document exists, this document is controlling as far as this Standard is concerned.

The physical layer of a DMX512 data link is constrained by earth grounding practices, termination methods, signal levels, EMC, and accidental damage by connection to other devices.

In addition to complying with the requirements of EIA-485-A, clauses 5 and 6 specify additional requirements not addressed by EIA-485-A.

4.2 Electrical isolation

EIA-485-A makes no general provisions for electrical isolation, however, this Standard does. Suitable optical isolation, transformer isolation, or other means may be employed to prevent the undesirable propagation of voltages that exceed the Common Mode limits of EIA-485-A (see clauses 5 and 6). The inclusion of such isolation does not, however, alter the requirement that a transmitter or receiver conforms to EIA-485-A.

4.3 Topology

A data link shall consist of a single active differential line driver, a terminated transmission line and one or more differential line receivers meeting the requirements of EIA-485-A and all additional requirements of this Standard. A DMX512 controlled device capable of returning status information is referred to as a responder in this Standard. For the purposes of this Standard, responders are considered to fall under the category of receiving devices. This subject is covered in Annex B.

4.4 DMX512 ports

A DMX512 Port is the signal connection point between the internal electronics of a device and the physical transmission line (cable), referred to as the equipment interconnect points by EIA-485. It may be made either by the prescribed connector as defined in clause 7 or by a terminal strip. A DMX512 port carries four signals and a common reference, designated data 1-, data 1+, data 2-, data 2+, and data link common. (In some situations, data 2- and data 2+ are not connected /provided). Historically, these signals

have been referred to as Pins which correlates to the physical pinout used on the XLR style connectors as defined in clause 7 of this Standard. There are other situations where different physical connections may be encountered, such as a terminal strip.

4.5 Data link common and grounding topologies

Various portions of clause 5 and Annex A deal with shield-to-earth ground topologies. In all cases there is a low impedance connection between data link common pin or contact of the DMX512 port and signal common of the EIA-485-A driver or receiver circuitry.

4.6 Preferred method of earth grounding data link common

DMX512 systems should make use of earth ground referenced transmitting devices and isolated receiving devices. This approach provides for a single point solid ground/chassis connection at the source, and allows for variations in building ground potentials between transmitting and receiving devices. This is to ensure that interoperability of equipment is achieved in situations that might otherwise exceed the Common Mode limitations of EIA-485-A. See EIA-485-A clause 4.3.1. Other approaches are covered in Annex A.

4.7 Primary data link

Data 1- and Data 1+ of a DMX512 Port form the primary data link. Format of the data is covered in clause 8. Limited use of multiple data link drivers for half-duplex, bi-directional data transmission on the primary data link is permitted in accordance with Enhanced Functionality as described in Annex B.

4.8 Secondary data link

Implementation of the secondary data link is optional.

4.8.1 Secondary data link - active use

Data 2- and Data 2+ of a DMX512 Port provide a secondary EIA-485-A data link. Implementation of this data link is optional. Several different network topologies are associated with the implementation of the secondary data link. Use of the secondary data link is permitted only in accordance with Enhanced Functionality as described in Annex B.

4.8.2 Secondary data link - passive loop through ports

In order to extend Enhanced Functionality across a network, devices containing two DMX512 ports, one for receive and one for transmit, that do not actively process or buffer data, shall provide a direct passive link for all signals between the two ports. Devices containing three or more DMX512 ports may provide a passive link between only two of the ports. These two ports shall be declared as required in Clause 10 and should be marked on the product. In order to enhance interoperability of DMX512 products, equipment designers are encouraged to provide passive loop through on the secondary data link pins or contacts whenever possible, even if not required per this clause.

4.9 Data link termination procedures

DMX512 data links shall be terminated to eliminate ringing and signal reflection, which can cause mis-operation of an otherwise properly designed system. To comply with this Standard, all equipment connected to a DMX512 data link shall operate in accordance with the stated manufacturer's specification when the data link is terminated. The terminator shall be a 120 ohm +5%/-10% impedance placed between Data+ and Data-. In the preferred topology where the transmitter is connected to one end of the data link, the far end of the data link shall be terminated. In the case where the transmitter cannot be connected at one end of the data link, then both ends of the data link shall be terminated. Manufacturers of receiving devices may provide internal termination of the data link. Where such termination is provided, it shall comply with the electrical and marking requirements of this Standard. It is recommended that termination components be chosen to withstand continuous voltages of at least 30 VAC 50 Hz/42 VDC.

4.10 Unpowered devices

Unpowered connected DMX512 devices shall not degrade the performance of the DMX512 transmission system nor materially lower the impedance they present to the data link.

5 Nominal Operating Characteristics

5.1 General

Operation limits generally follow the detailed requirements of EIA-485-A. Where appropriate, separate limits are given for isolated products. All electrical characteristics shall be measured at the DMX512 ports of the product.

5.2 Chassis in power isolated equipment

In equipment where connection to protective grounding is not made concurrent with power connection, such as battery powered equipment or equipment powered by isolated low voltage transformers, chassis is deemed to include any exposed metal DMX512 connector parts which do not carry signals.

5.3 Earth grounding of data link common for transmitters

In recognition of the need for DMX512 compliant products to be capable of interconnection as part of large and potentially complex systems, this Standard defines two allowable topologies for the earth grounding of data link common and circuit common for transmitters, to be known as "Ground Referenced" and "Isolated". The preferred method is "Ground Referenced." "Isolated" is covered in Annex A.

5.4 Ground referenced transmitters

Ground referenced transmitting device outputs shall meet the following conditions in table 1 during normal operation under open circuit condition.

Table 1 - Ground Referenced Transmitter Characteristics

Connection	Limit (measured at port pin or contact)	Comment
Data 1+ to Data Link Common or Data 1- to Data Link Common	$0 \leq v \leq +6 \text{ VDC}$	
Data 2+ to Data Link Common or Data 2- to Data Link Common	$0 \leq v \leq +6 \text{ VDC}$	Enhanced Function devices only
Data Link Common to Chassis	0V	
Data 1- to Chassis or Data 1+ to Chassis	$0 \leq v \leq +6 \text{ VDC}$	
Data 2- to Chassis or Data 2+ to Chassis	$0 \leq v \leq +6 \text{ VDC}$	Enhanced Function devices only
Data 1- to Data1+	+/- 6V (open circuit)	
Data 2- to Data 2+	+/- 6V (open circuit)	

Figure 1 illustrates a ground referenced transmitter port. It is characterized by the direct connection of the shield (Data Link Common) to chassis and protective earth. Therefore, devices employing ground referenced transmitters shall be provided with provision for connection to protective earth. Any resistance (A) between Data Link Common's pin or contact and zero volt supply (circuit common) of the transmitter circuit shall be less than 100 ohms. Any resistance (B) between Data Link Common's pin or contact and chassis shall be less than 20 ohms and is preferably zero ohms.

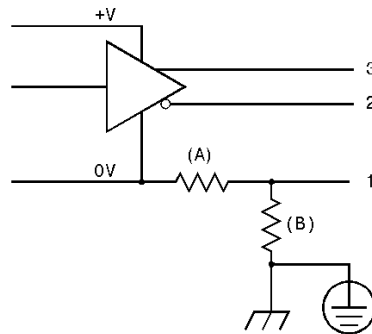


Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- A - Optional Resistance (see text)
- B - Optional Resistance (see text)

Figure 1 - Ground Referenced Transmitter

A DMX512 device may have any number of Ground Referenced transmitter ports. Ground Referenced transmitter ports may be used by all DMX512 devices including ones that provide any number of non-DMX512 input or output ports. Adherence to this topology allows a DMX512 transmitter connector to be marked as shown in table 9.

Because the transmitter in this topology is grounded, the existence of an isolation barrier between the transmitter and any other part of the device shall NOT qualify output for marking as ISOLATED.

5.5 Disallowed transmitter topology

The Figure 2 configuration is not permitted, although it may exist on some legacy products. While this topology is described as one possible topology in EIA-485-A, it is not appropriate when considering operation of DMX512 transmitting devices in systems encountering differential ground potentials.

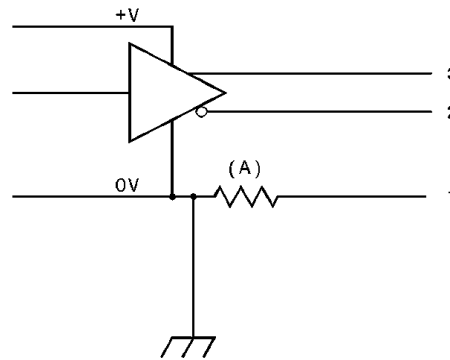


Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- A - $R > 0.2 \text{ ohm}$

Figure 2 - Transmitter Topology NOT Allowed

5.6 Earth grounding of data link common for receivers

This Standard defines several allowable topologies for earth grounding of data link common and circuit common for receiving devices. These are to be known as “non-isolated” and “isolated”. The preferred method is “isolated.” A specific concession is available to manufacturers of non-isolated receivers who, for reasons beyond the scope of this Standard, require a direct link between data link common and chassis. Non-isolated and grounded receivers are addressed in Annex A.

5.7 Isolated receiver characteristics

For isolated devices, a capacitor may be fitted between Data Link Common and chassis for the purpose of Radio Frequency bypass. Devices shall continue to operate correctly when exposed to any of the conditions in table 2.

Table 2 - Isolated Receiver characteristics

Connection	Limit (measured at port pin or contact)	Comment
Data 1- to Data Link Common or Data 1+ to Data Link Common	+12 / -7 VDC	Common Mode range
Data 2- to Data Link Common or Data 2+ to Data Link Common	+12 / -7 VDC	Enhanced Functionality devices only
Data Link Common to Chassis	$\geq 22\text{M ohm @ } 42 \text{ VDC}$	
Data 1- to Chassis or Data 1+ to Chassis	$\geq 22\text{M ohm @ } 42 \text{ VDC}$	
Data 2- to Chassis or Data 2+ to Chassis	$\geq 22\text{M ohm @ } 42 \text{ VDC}$	Enhanced Functionality devices only
Data 1- to Data 1+	+/- 6 V	
Data 2- to Data 2+	+/- 6 V	Enhanced Functionality devices only
Any Port Pin or Contact to Chassis	30 VAC / 42 VDC	

Figure 3 illustrates an isolated receiver. Any signal pin or contact of the DMX512 isolated receiver shall present a resistance (B) greater than 22 Mohm at 42 VDC with respect to Chassis, with respect to

Protective Ground (where fitted), with respect to any other signal inputs or outputs, and with respect to other ground referenced electronics. There may be a capacitance (not shown) between Data Link Common and chassis for Radio Frequency bypass. Any resistance (A) between Data Link Common's pin or contact and zero volt supply (circuit common) supply of the receiver circuit shall be less than 100 ohms.

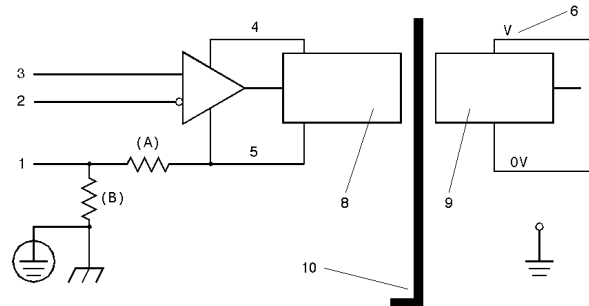


Figure Key

- | | |
|---------------------------------|---------------------------------------|
| 1 - DMX512 Data Link Common | 9 - Optional Non-Isolated Electronics |
| 2 - DMX512 Data 1- (or Data 2-) | 10 - Isolation Barrier |
| 3 - DMX512 Data 1+ (or Data 2+) | A - Optional resistance (see text) |
| 4 - Isolated Supply | B - Optional resistance (see text) |
| 5 - Isolated 0V Supply | |
| 6 - V (+ or -) | |
| 7 - I / O | |
| 8 - Isolated Electronics | |

Figure 3 - Isolated Receiver

Adherence to this topology allows a DMX512 receiver port to be marked as ISOLATED. Isolated receiver ports may be used by all DMX512 devices including ones that provide any number of non-DMX512 input or output ports. A DMX512 device may have any number of isolated receiver ports

5.8 Disallowed receiver topology

Figure 4 configuration is not permitted, although it may exist on some legacy products. While this topology is described as one possible topology in EIA-485-A, it is not appropriate when considering operation of DMX512 receiving devices in systems encountering differential ground potentials.

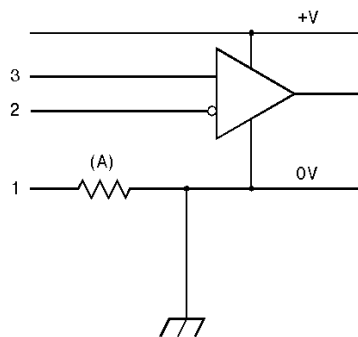


Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- A - $R \geq 0.2 \text{ ohm}$

Figure 4 - Receiver Topology NOT Allowed

5.9 DMX512 processing devices

It is permissible to design processing devices based on the Isolated Receiver / Ground Referenced Transmitter or Isolated Receiver / Isolated Transmitter models (see Annex A) already described.

5.10 Loading designation

As per EIA-485-A, the total load permitted on a DMX512 data link is 32 unit loads. Transmitters designed for this Standard shall be capable of driving 32 unit loads on a DMX512 data link. Each receiving port on a DMX512 device shall have a unit load of 1 or less as per EIA-485-A receiver biased to any voltage from -7 to +12 volts shall not present a capacitive load to the line of more than 125 pF per unit load.

Declaring or marking of the unit load is not required by this Standard. If a manufacturer chooses to declare or mark their products with a unit load value, the declared or marked value shall be the greater of either the DC unit load determined by EIA-485-A clause 4.1 or the unit load as determined by the capacitive loading. In either case, if the unit load is declared the capacitive load values shall also be declared.

6 Protection

6.1 Minimum protection against interconnection damage

Clause 4.2 of EIA-485-A recognizes that certain other extraneous conditions may overstress the system and that these conditions should be specified in the referencing standard. Extensive use of temporary and portable equipment in Entertainment Lighting Industry results in frequent connection and disconnection of equipment and gives rise to the possibility of equipment misconnection.

Equipment may be protected against damage resulting from accidental connection to voltages in excess of the minimum defined in EIA-485-A clauses 4.2.5 and 4.2.6, and this is recommended. See Annex C for requirements allowing DMX512 ports to be declared "DMX512-A Protected". This does not negate the need to comply with EIA-485-A clause 4.2.6 - Transient overvoltage tolerance.

6.2 Minimum Electro Static Discharge (ESD) protection

Manufacturers shall ensure that any port pin or contact can withstand a minimum of 4kV ESD for contact discharge and 8kV ESD for air discharge in accordance with IEC 61000-4-2 or other local regulations that may require higher levels of protection. The acceptance criteria for this requirement allows for temporary loss of function, provided the function is self recoverable or can be restored by the operation of controls. Meeting this requirement does not alter the fact that a manufacturer may have to meet other more stringent ESD requirements to conform to local EMC regulations.

7 Connection Methods

7.1 Equipment fitted with user accessible pluggable data link connections

This category includes all portable products.

Female connectors shall be used on controllers and other transmitting devices (primary data link) and male connectors shall be used on receiving devices. Female and male connectors consistent with this convention shall also be used where loop-through connections are provided.

7.1.1 Required connector

Equipment in this category shall use 5-pin XLR connectors with the physical pinout of the 5-pin XLR in accordance with Table 3.

Table 3- Signal designations summary

Use	5-Pin XLR Pin #	DMX512 Function
Common Reference	1	Data Link Common
Primary Data Link	2	Data 1-
	3	Data 1+
Secondary Data Link (Optional - see clause 4.8)	4	Data 2-
	5	Data 2+

7.1.2 Concession for use of an alternate connector (NCC DMX512-A)

A concession to use an alternate connector is available only when it is physically impossible to mount a 5-pin XLR connector on the product. In such cases all the following additional requirements shall be met :

- 1) The alternate connector shall not be any type of XLR connector.
- 2) The alternate connector shall not be any type of IEC 60603-7 8-position modular connector except as allowed in clause 7.3.
- 3) Provided that all other requirements of this Standard are met, in addition to the declaration in the equipment manual, the alternate connector shall be marked as NCC DMX512-A (Not Connector Compatible). If such a marking is not physically possible at the connector, an appropriate marking shall be made elsewhere on the equipment. The pin numbering on the alternate connector should match numbering for the standard 5-Pin XLR connector.
- 4) The manufacturer shall make available an adapter with the appropriate connections to a standard 5-pin XLR connector for all DMX512 ports included in the alternate connector.
- 5) The Enhanced Functionality, if applicable, and ground/isolation declarations shall continue to be declared for each DMX512 port.

7.2 Equipment intended for fixed installation with internal connections to the data link

Fixed installation products with internal connections to the data link may use the 5-pin XLR connector, but shall not use any other XLR connector. When use is made of the 5-pin XLR connector, the requirements of 7.1 and 7.1.1 shall apply. When a non-XLR connector is used, this Standard makes no other restriction or stipulation on connector choice. The contact (pin) numbering on the alternate connector should match numbering for the standard 5-Pin XLR

7.3 IEC 60603-7 8-position modular connectors

The use of IEC 60603-7 8-position modular connectors (commonly referred to as RJ45 type connectors – plugs/jacks) and associated punchdown terminal blocks shall be limited to connections that are part of a fixed installation and not normally accessible except to qualified, authorized users, nor intended for regular connection and disconnection. External (user accessible) IEC 60603-7 8-position modular connectors are permitted only on patch and data distribution products and only when permanently installed in controlled access areas.

Note: Examples of not normally accessible or controlled access areas include a locked electrical room or control booth, provided those who need access have a key (or lock combination) available.

Table 4 - Connection Schedule for DMX512 Equipment Using IEC 60603-7 8-Position Modular Connectors

Pin (Wire) #	Wire Color	DMX512 Function
1	white / orange	data 1+
2	orange	data 1-
3	white / green	data 2+ (optional)
6	green	data 2- (optional)
4	blue	Not assigned
5	white / blue	Not assigned
7	white / brown	Data Link Common (Common Reference) for Data 1 (0 v)
8	brown	Data Link Common (Common Reference) for Data 2 (0 v)
	drain	

Note 1: Pin numbering and color in accordance with ANSI/TIA/EIA-568 scheme T568B.

Note 2: Pin 8 should be wired as signal common even if pins 3 and 6 are NOT wired so that both conductors 7 and 8 are at equal potential.

Warning: Accidental connection to non-DMX512 equipment likely to be encountered (e.g., an Ethernet Hub at a patch bay) may result in damage to equipment. Pins 4 and 5 may carry voltages outside the EIA-485 range in telecom applications (e.g., telephone ringing). Pins 4 and 7 may carry voltages outside the EIA-485 range in other applications (e.g., some manufacturers whose distributed DMX512 buffering products require low voltage DC power may use these wires for this purpose). Because of these various uses, misplugging unlike systems could cause serious damage.

8 Data protocol

8.1 Format

DMX512 slots shall be transmitted sequentially in asynchronous serial format, beginning with slot 0 and ending with the last implemented slot, up to slot 512 (a maximum total of 513 slots). Prior to the first data slot being transmitted, a Reset Sequence shall be transmitted – a BREAK, followed by a MARK AFTER BREAK, and a START Code. Valid DMX512 data slot values under a NULL START Code shall be 0 to 255 decimal.

8.2 Slot format

The data transmission format for each data value transmitted are as shown in table 5. Note that no parity is transmitted.

Table 5 - Data Slot Format

Bit Position	Description
1	Start Bit, Low or SPACE
2 through 9	Slot Value Data Bits, Least Significant Bit to Most Significant Bit
10, 11	Stop Bits, High or MARK

8.3 Break

The BREAK indicates the start of a new packet.

The BREAK generated by a transmitter is defined as a mark-to-space transition followed by a low of at least the minimum duration shown in table 6 (Timing Diagram, Designation #1) followed by a low to high transition.

Table 7 (Timing Diagram, Designation 1) defines the minimum duration break a receiver is required to recognize as the start of a new packet.

8.4 Mark after break

The MARK separating the BREAK and the START Code (Timing Diagram, Designation #2) is defined as the MARK AFTER BREAK. DMX512 transmitters produce a MARK AFTER BREAK that complies with the minimum and maximum values for designator 2 in table 6. Compliant receivers correctly respond to data streams with a MARK AFTER BREAK at least as long as the minimum shown in table 7.

Note: The 1986 version of this Standard specified a 4 microsecond MARK AFTER BREAK. The 1990 version of the standard changed that value to 8 microseconds, but added an option for receivers capable of recognizing the 4 microsecond MARK AFTER BREAK to be identified as having that capability. Some transmitters may still be in use that generate the shorter 4 microsecond MARK AFTER BREAK, and they may not work with equipment built to this Standard.

8.5 START code

The START Code is the first slot (slot 0) following a MARK AFTER BREAK. The START Code identifies the function of subsequent data in that packet.

8.5.1 NULL START code

A NULL START Code identifies subsequent data slots as a block of un-typed sequential 8-bit information. Packets identified by a NULL START Code are the default packets sent on DMX512 net-works. Earlier versions of this Standard assumed that only dimmer class data would be sent using NULL START Code packets. In practice NULL START Code packets have been used by a wide variety of devices; this version recognizes this fact.

Each NULL START Code packet contains no formal data or addressing structure. The device using data from the packet must know the position of that data within the packet.

There is no guarantee that all NULL START Code packets will be delivered to all devices. Data sent using NULL START Codes should be of a type where loss of packet does not greatly affect the operation of the device. Hence data sent should be of the current value of a parameter, not a command to execute a routine. Once a controller is configured for a particular application, all NULL START Code packets should have the same number of slots.

8.5.2 Dimmer class data

Dimmer level data should be sent in NULL START Code packets. Valid dimmer levels shall be 0 to 255 decimal (00 to FF hexadecimal) representing dimmer control input. Value 0 shall represent a dimmer output of OFF or minimum and 255 shall represent an output of FULL. A dimmer shall respond to increasing the DMX512 slot value for 0 to 255 by fading from its minimum level (off) to its maximum level

(full). The exact relationship between DMX512 slot values and dimmer output is beyond the scope of this Standard.

Note that NULL START Code packets are the default packets sent over DMX512 networks and may contain data other than dimmer class data.

8.5.3 Other START codes

In order to provide for future expansion and flexibility, DMX512 makes provision for 255 additional non NULL START Codes (1 through 255 decimal, 01 through FF hexadecimal), henceforth referred to as Alternate START Codes. Where it is required to send proprietary information over a DMX512 data link, a packet starting with a registered Alternate START Code shall be used.

Several Alternate START Codes are reserved. See Annex D.

See Annex E for Alternate START Code Registration Policies.

8.5.3.1 Alternate START code refresh interval

A DMX512 transmitter interleaving NULL START Code packets with Alternate START Code packets shall send a NULL START Code packet at least once per second.

8.5.3.2 Timing differences for Alternate START code packets

To ensure that in-line processing devices do not lose essential Alternate START Code data, a reduction of the maximum Alternate START Code update rate is necessary. One of the following methods shall be used:

- 1) Alternate START Code packets may be transmitted slower than the minimum timings specified in clause 8.11 by increasing the break to break time to 10% more than the minimum required for the Alternate START Code packets number of slots.
- 2) Transmitters may interlace non essential NULL START Code packets with Alternate START Code packets.

8.5.3.3 Handling of Alternate START code packets by in-line devices

DMX512 processing devices or any device that receives and re-transmits DMX512 shall state in the manual for the product how they process Alternate START Code packets. The acceptable processing methods are:

- 1) Block all packets containing particular Alternate START Codes. The START Codes blocked shall be declared (and may be all Alternate START Codes).
- 2) Pass unmodified all packets containing particular Alternate START Codes. The START Codes passed shall be declared.
- 3) Process the information contained in packets containing particular Alternate START Codes. The algorithm shall be declared in enough detail to allow the user to decide if the device will satisfy their needs.

DMX512 in-line repeating transmitters shall not pass some packets with a particular Alternate START Code while blocking other packets containing the same Alternate START Code unless doing so as part of a stated processing algorithm.

8.5.4 START code processing

All receiving devices other than in-line processing devices shall process the START Code and differentiate between those packets with NULL START Codes and those with Alternate START Codes. Devices shall not ignore START Codes by assuming that all packets received are NULL START Code packets.

8.6 Maximum number of data slots

Each data link shall support up to 512 data slots. Multiple links shall be used where larger numbers of slots are required.

8.7 Minimum number of data slots

There shall be no minimum number of data slots on the data link. DMX512 data packets with fewer than 512 slots may be transmitted, subject to the minimum timing requirements of this Standard – see clause 8.10 and figure 5.

8.8 Defined line state between slots

The time between any two slots of a data packet (Timing Diagram, Designation #9) may vary between minimum and maximum values shown for designator #9 of table 6. The line shall remain in a marking state during any such idle period.

8.9 Defined line state between data packets (Mark Before Break)

Every data packet transmitted on the data link, regardless of START Code or length, begins with a BREAK, MARK AFTER BREAK, and START Code sequence known as a Reset Sequence (Timing Diagram - Figure 5, Item 12). The time between the second stop bit of the last data slot of one data packet and the falling edge of the beginning of the BREAK for the next data packet (Timing Diagram, Designation #10) may vary between minimum and maximum values shown for designator #10 of table 6. The line shall remain in a marking state throughout any such period. Transmitters, therefore, shall not produce multiple BREAKs between data packets. Receivers, however, shall be capable of recovering from multiple BREAKs produced by data link line errors.

8.10 Break-to-Break spacing

Transmitters produce packets so that the period between the falling edge at the start of any one BREAK and the falling edge at the start of the next BREAK are not be less than the minimum value in table 6 designator #13. This period is also not more than the maximum value in table 6 designator #13.

Receivers operate correctly when receiving packets with break to break spacing of at least the minimum value in table 7 designator #13 up to the maximum value in table 7 designator #13.

8.11 Timing Diagram - data+

Timings shall follow the requirements of the timing diagram (figure 5) and its associated tables 6 and 7.

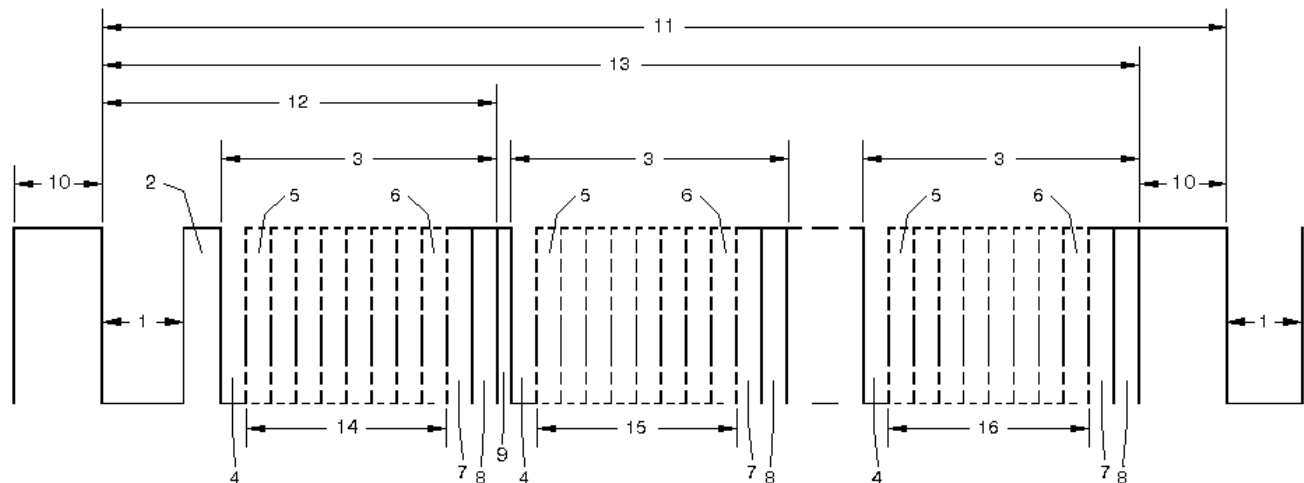


Figure Key

- 1 - "SPACE" for BREAK
- 2 - "MARK" After BREAK (MAB)
- 3 - Slot Time
- 4 - START Bit

- 5 - LEAST SIGNIFICANT Data BIT
- 6 - MOST SIGNIFICANT Data BIT
- 7 - STOP Bit
- 8 - STOP Bit
- 9 - "MARK" Time Between slots
- 10- "MARK" Before BREAK (MBB)
- 11- BREAK to BREAK Time
- 12- RESET Sequence (BREAK, MAB, START Code)
- 13- DMX512 Packet
- 14- START CODE (SLOT 0 Data)
- 15- SLOT 1 DATA
- 16- SLOT nnn DATA (Maximum 512)

Figure 5 - Timing Diagram

Table 6 - Timing Diagram Values - output of transmitting UART

Designation	Description	Min	Typical	Max	Unit
-	Bit Rate	245	250	255	kbit / s
-	Bit Time	3.92	4	4.08	μs
-	Minimum Update Time for 513 slots	—	22.7	—	ms
-	Maximum Refresh Rate for 513 slots	—	44	—	updates / s
1	"SPACE" for BREAK	92	176	—	μs
2	"MARK" After BREAK (MAB)	12	—	< 1.00	μs s
9	"MARK" Time between slots	0	—	< 1.00	s
10	"MARK" Before BREAK (MBB)	0	—	< 1.00	s
11	BREAK to BREAK Time	1204	—	— 1.00	μs s
13	DMX512 Packet	1204	—	— 1.00	μs s

Table 7 - Timing Diagram Values for Receivers

Designation	Description	Min	Typical	Max	Unit
-	Bit Rate	245	250	255	kbit / s
-	Bit Time	3.92	4	4.08	µs
-	Minimum Update Time for 513 slots	–	22.7	–	ms
-	Maximum Refresh Rate for 513 slots	–	44	–	updates / s
1	“SPACE” for BREAK	88	176	–	µs
2	“MARK” After BREAK (MAB)	8	–	< 1.00	µs s
9	“MARK” Time between slots	0	–	< 1.00	s
10	“MARK” Before BREAK (MBB)	0	–	< 1.00	s
11	BREAK to BREAK Time	1196	–	– 1.25	µs s
13	DMX512 Packet	1196	–	– 1.25	µs s

Note: Given the range of timing parameters shown in tables 6 and 7, a transmitter may produce a signal with a refresh rate in the range of 1 Hz to approximately 830 Hz. A receiver must be capable of accepting a signal with a refresh rate of 0.8 Hz to approximately 836 Hz.

9 Receiver Performance

9.1 Rejection of Improperly framed slots

A receiver shall check the first stop bit and should check the second stop bit of all received slots to determine if they have the correct value. If a missing stop bit is detected, the receiver shall discard the improperly framed slot data and all following slots in the packet.

9.2 Loss of data tolerance / Resumption of acceptance of data

A receiver not receiving a Reset Sequence (a sequence of a Break, Mark After Break, and START Code) within one maximum break to break time (as per table 7) of the previous Reset Sequence is considered to have lost data input.

Although this Standard does not specify loss of data handling procedures, manufacturers shall state what their Loss of Data handling procedures are.

Note: In the absence of overriding safety issues or an alternative source of control data, when encountering a loss of data condition a receiving device should remain in an operating condition for at least 60 seconds, awaiting resumption of the DMX512 signal.

9.3 Receiver performance at maximum refresh rate

Any device incorporating a DMX512 receiver shall operate correctly when receiving continuous transmission of valid data packets containing any number of valid slot values.

9.4 Packet processing latency

Some products may provide their specified functionality without processing or being able to process every consecutive DMX512 packet. Such products will have an inherent latency to data changes between packets, which shall be declared by the manufacturer in accordance with the disclosure requirements of clause 10.

10 Marking and Disclosures

In this Standard, marking is defined as the application of a physical mark/label, etc. on the product, and declaration means declared in a manual and optionally marked on the product.

10.1 Identification

Only equipment conforming to this Standard may be marked and identified with "USITT DMX512-A" or "DMX512-A".

10.2 DMX512 port marking

Where required by clauses of this Standard, all DMX512 ports shall be marked as to the applicable Enhanced Functionality as defined in table B1 (Annex B). All information provided by marking shall also be provided in the equipment manual.

Manufacturers shall use the signal designations from table 3 in any pinout detail declaration or marking of pin, contact or terminal functions appearing on or within a product and associated with installation or connection. Where it is necessary to use abbreviations, only those detailed in table 8 shall be permitted.

Table 8 - Signal designations abbreviations allowed for marking

Function	Preferred Abbreviation	Non preferred Abbreviation
Common Reference	COM	C
Primary Data Link – Data 1-	D1-	1-
Primary Data Link – Data 1+	D1+	1+
Secondary Data Link – Data 2-	D2-	2-
Secondary Data Link – Data 2+	D2+	2+




10.3 Data line termination marking

Products that have switchable (in/out) termination circuits shall provide indication of the termination state.

10.4 Ground / Isolation marking

All DMX512 ports shall indicate the relationship between Data Link Common and earth ground. The allowed grounding topologies are shown in table 9.

Table 9 - Ground / Isolation marking

Function of DMX512 port	Defining Clause / Figure	Comment	Approved Marking
Transmitter	clause 5.4 / fig 1	Ground Referenced	no mark required, recommended mark 
Transmitter	clause A1 / fig A1	Isolated (See Annex A)	ISO or ISOLATED
Transmitter	clause A4 / fig A4	Floating (See Annex A)	FLT or FLOAT or FLOATING
Receiver	clause A2 / fig A2	Non-Isolated, 100 ohm 2 Watt resistor	NON-ISO or NON-ISOLATED
Receiver	clause A3 / fig A3	Grounded – concession per clause A3	 
Receiver	clause 5.7 / fig 3	Isolated	<i>no mark required</i> ; ISO or ISOLATED
Receiver	clause A4 / fig A4	Floating	FLT or FLOAT or FLOATING

Note: Although no mark is required of Ground Referenced transmitters or Isolated receivers, it is highly recommended that such markings appear on the product.

10.5 Required disclosures and markings

10.5.1 Portable products and products fitted with external pluggable data link connectors

DMX512 ports on these products shall be marked in accordance with clause 10.2. DMX512 ports on these products shall provide Ground/Isolation marking in accordance with clause 10.4.

If use has been made of any non-XLR connector in conjunction with the supply of an adapter as permitted by clause 7.1.2, the non-XLR connector shall be declared in the equipment manual as NCC DMX512-A and also be marked (when feasible) in accordance with that clause.

10.5.2 Equipment intended for fixed installation with internal connections to the data link

DMX512 ports on these products shall be marked in accordance with clause 10.2. DMX512 ports on these products shall provide Ground/Isolation marking in accordance with clause 10.4.

Clearly identified terminal contact (connector pinout) detail shall be marked on or within any product in accordance with clause 10.2.

10.5.3 Loss of data handling procedure

This Standard does not define loss of data handling procedures (clause 9) beyond requiring that manufacturers declare their own products' procedure(s). Such declarations shall be made in the equipment manual.

10.5.4 Packet processing latency

Manufacturers of in-line DMX512 processing, or distribution devices, shall declare any inherent latency in the processing of data packets, including a statement of the worst case delay between receiving a packet and that packet affecting the product's output. A statement of the worst case percentage number of zero start code packets that are discarded by the product shall also be included.

10.5.5 NULL START Code functionality

Manufacturers of transmitting devices shall declare in the equipment manual the full range of slot values transmitted in conjunction with packets sent using the NULL START Code.

Manufacturers of receiving devices shall declare the response to packets received containing the NULL START Code, with particular reference to any functionality requiring limited or restricted slot data values.

10.5.6 Slot footprint

Manufacturers of receiving devices shall declare the slot footprint in the equipment manual.

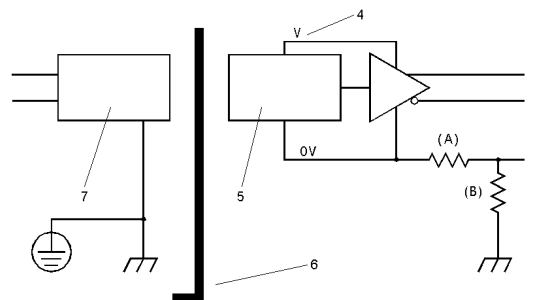
= END =

Annex A - Non Preferred (Alternate) topologies (Normative)

A1 Isolated transmitters

This standard permits the use of isolated transmitter ports. They are used in legacy systems where grounded receivers are used and large Common Mode voltages are expected.

Figure A1 illustrates an isolated transmitter port. To be considered an isolated transmitter, any signal pin or contact of the DMX512 output shall present a resistance (B) greater than 22 Mohm at 42 VDC with respect to Chassis, and with respect to Protective Ground (where fitted). If there are any other signal inputs or outputs, any signal pin or contact of the DMX512 output shall present a resistance greater than 22 Mohm at 42 VDC with respect to these inputs or outputs. The power supply for the transmitter and any directly connected electronics shall be isolated from earth ground and any other grounded referenced electronics to levels at least as high as those required of the output. Any resistance (A) between Data Link Common's pin or contact and zero volt supply (circuit common) of the transmitter circuit shall be less than 100 ohms. There may be a capacitance (not shown) between Data Link Common and chassis for the



purpose of Radio Frequency bypass.

Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+(or Data 2+)
- 4 - V (+ or -)
- 5 - Other Isolated Electronics
- 6 - Isolation Barrier
- 7 - Non-Isolated Electronics (optional)
- A - Optional resistance (see text)
- B - Resistance (see text)

Figure A1 - Isolated Transmitter

Table A1 - Isolated Transmitter characteristics

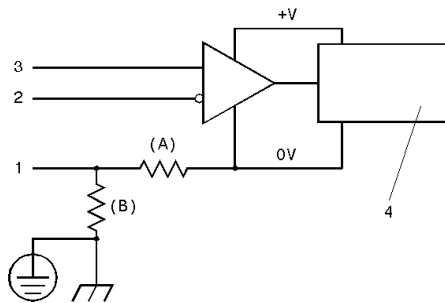
Connection	Limit (measured at port pin or contact)	Comment
Data 1- to Data Link Common or Data 1+ to Data Link Common	+/- 6 V	Common Mode range
Data 2- to Data Link Common or Data 2+ to Data Link Common	+/- 6 V	Enhanced Function devices only
Data Link Common to Chassis	$\geq 22\text{M ohm @ 42 VDC}$	
Data 1- to Chassis or Data 1+ to Chassis	$\geq 22\text{M ohm @ 42 VDC}$	
Data 2- to Chassis or Data 2+ to Chassis	$\geq 22\text{M ohm @ 42 VDC}$	Enhanced Function devices only
Data 1- to Data 1+	+/- 6 V	
Data 2- to Data 2+	+/- 6 V	Enhanced Function devices only
Any Port Pin or Contact to Chassis	30 VAC / 42 VDC	

A DMX512 device may have any number of isolated transmitter ports.

Adherence to this topology allows a DMX512 transmitter connector to be marked ISO or ISOLATED.

A2 Non-isolated receivers

While not the preferred topology, this non-isolated topology exhibits considerable improvement in common mode fault tolerance compared to other grounded or non-isolated topologies. In this topology there shall be a resistance (B) of 100 ohms between Data Link Common's pin or contact and chassis. This resistance shall be able to safely dissipate two watts. No other connection between data link common and chassis is permitted. Common Mode voltage present between the chassis of this device and the chassis of any other device connected to the DMX512 shield will cause a current flow. This current will cause a voltage drop in the resistance (B). This voltage drop effectively decreases the Common Mode voltage at this receiver. Any resistance (A) between Data Link Common's pin or contact and zero volt supply (circuit



common) of the receiver circuit shall be less than 100 ohms.

Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- 4 - Other Electronics
- A - Optional resistance (see text)
- B - Optional resistance (see text)

Figure A2 - Non-Isolated Receiver

Since 0 V is not directly referenced to Chassis, local product safety standards may restrict choice of power supply (e.g., use of a Class 2 supply).

A DMX512 receiver may have any number of non-isolated receiver ports. Where multiple ports are implemented, the total parallel resistance (B) shall be 100 ohms.

Table A2 - Non-Isolated Receiver characteristics

Connection	Limit	Comment
Data 1- to Data Link Common or Data 1+ to Data Link Common	+12 / -7 VDC	Common Mode range
Data 2- to Data Link Common or Data 2+ to Data Link Common	+12 / -7 VDC	Enhanced Function devices only
Data Link Common to Chassis	100 ohms	-Note 1-
Data 1- to Chassis or Data 1+ to Chassis	+12 / -7 VDC	
Data 2- to Chassis or Data 2+ to Chassis	+12 / -7 VDC	Enhanced Function devices only
Data 1- to Data 1+	+/- 6V	
Data 2- to Data 2+	+/- 6V	Enhanced Function devices only
Any Port Pin or Contact to Chassis	N/A	

Note 1: This cannot be characterized in terms of voltage.

Manufacturers shall be permitted to fit a resistance of 100 ohms +/-20% between Chassis and Data Link Common for the purpose of limiting current in the shield due to small differential ground potentials. This method provides for reduction of Common Mode voltage at the line receiver.

The only output that shall be permitted to be directly connected to this topology is a single passive DMX512 loop through port. No other inputs or outputs are allowed with this topology unless they meet the requirements of clause A4.

A3 Grounded Receivers

The topology of Figure A3 is allowed for the construction of entry level receivers where the cost of isolation might prove an untenable burden. It may be used by manufacturers of receivers who, for reasons beyond the scope of this Standard require a direct link between data link common and protective earth. It is not a recommended practice and requires special marking on the product and special explanatory text in all manuals.

This topology is characterized by the direct connection of the shield (Data Link Common) to chassis and protective earth. Therefore, devices employing ground referenced receivers shall be provided with a connection to protective earth. Any resistance (A) between Data Link Common's pin or contact and zero volt supply (circuit common) of the receiver circuit shall be less than 100 ohms.

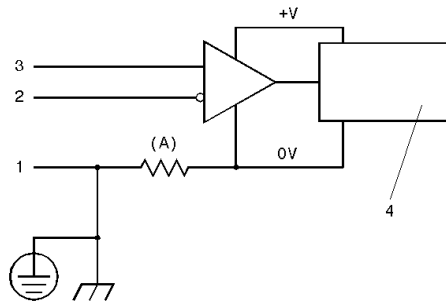


Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- 4 - Other Electronics
- A - Optional Resistance (see text)

Figure A3 - Grounded Receiver

Ports using this topology shall be marked using the standard symbol representation as defined in clause 10.5 and be declared.

A4 Earth grounding of data link common for floating devices

Figure A4 illustrates a floating topology. Floating is both an input and an output topology and is an additional allowable topology. It is often confused with the isolated DMX512 topology. As with the isolated topology, any input or output signal pin or contact shall present a resistance (B) greater than 22 Mohm at 42 VDC with respect to Chassis, and with respect to Protective Ground (where fitted). It differs from an isolated topology in that there is no requirement of isolation between DMX512 inputs and outputs. The resistance between Data Link Common's pin or contact of the DMX512 input and Data Link Common's pin or contact of the DMX512 output shall be less than 0.2 ohms. There may be a capacitance (not shown) between Data Link Common and chassis for the purpose of Radio Frequency bypass. Any resistance (A) between any Data Link Common's pin or contact and the zero volt supply (circuit common) of the transmitter and receiver circuits shall be less than 100 ohms.

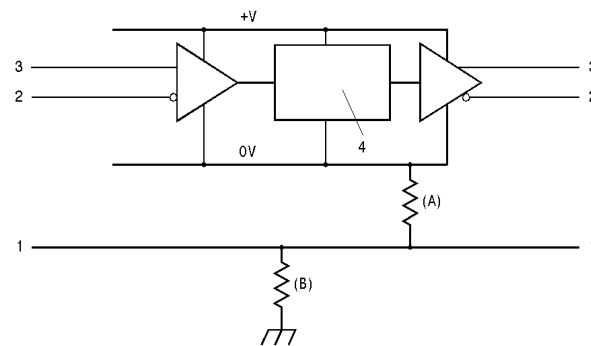


Figure Key

- 1 - DMX512 Data Link Common
- 2 - DMX512 Data 1- (or Data 2-)
- 3 - DMX512 Data 1+ (or Data 2+)
- 4 - Other Electronics
- A - Optional resistance (see text)
- B - Optional resistance (see text)

Figure A4 - DMX512 Device, Floating

The grounding of a device using this topology is determined by the connected devices. For that reason this topology shall not be used for devices that provide ground referenced non-DMX512 input or output ports unless those ports are isolated from DMX512 lines by an impedance of at least 22 Mohms.

Both input and output ports of a floating device shall be marked FLT, FLOAT, or FLOATING. A device may have only one floating DMX512 receiver port.

-end of Annex A-

Annex B (Normative) - Enhanced DMX512

B1 General

The original and 1990 versions of USITT DMX512 called out an “Optional Second Data Link.” There was no detailed guidance for its use. The majority of legacy systems did not use the second data link at all. Many uses of the second data link have been implemented over the years. While many of these were reasonable, a few uses clearly were not compliant with EIA-485-A. These uses vary in both their electrical requirements and in the data protocol used. One of the purposes of this Standard is to regularize the use of the second data link. It is no longer possible to select a single implementation and forbid all others. However, not all historical uses will be allowed to continue.

The network topologies needed to support Enhanced Functions are identified by an EF number. This edition of the standard supports four different EF topologies.

The introduction of standard EF topologies in no way changes backward compatibility of primary data link functions. In all cases, regardless of the EF topology, all DMX512 transmitters and receivers shall be able to interchange DMX512 data on the primary data link. All DMX512 devices shall be able to be connected without damage. All DMX512 devices shall not be damaged by connection to compliant legacy devices.

A DMX512 controlled device capable of returning status information is referred to as a responder in this Standard. For the purposes of this Standard, responders are not considered to fall under the category of transmitting devices.

B2 Summary of Enhanced Function Topologies

The Enhanced Function topologies are summarized in table B1. The data format on primary data link (Data 1- and Data 1+) shall adhere to the DMX512-A Standard

Table B1 - Enhanced Function topologies

EF #	Symbol (Optional)	Description	Comment
1	↔ —	half duplex EIA-485-A signals on primary data link; return signals on primary data link controlled by a registered Alternate START Code; no functionality on secondary data link	
2	→ ←	unidirectional DMX512 data on primary data link (EIA-485-A signals) and return EIA-485-A signals on secondary data link	
3	→ ↔	unidirectional DMX512 data on primary data link (EIA-485-A signals) and half duplex EIA-485-A signals on secondary data link	refer to manufacturers instructions
4	↔ ↔	half duplex EIA-485-A signals on both pairs; return signal on primary data link controlled by a registered Alternate START Code	

Note: References to unidirectional data are with respect to a transmitting device.

Different physical DMX512 ports on a product may be of different EF topologies and shall be declared and marked as required in clause 10.

B3 Identification of data protocols for Enhanced DMX512

The EF topology specifies the electrical and operating mode for the data links. It does not detail the data structure or protocol. The method of identifying data structures or protocols using the defined EF numbers is by way of an additional registered number placed after the EF number, separated from the EF number by a period. The registry for these designations will be maintained by Secretariat for the ANSI E1

Accredited Standards Committee – ESTA (see Annexes D and E). Numbers ending in zero are reserved for future development of the standard (e.g., EF1.0, EF1.10, EF2.20).

B3.1 EF1 Half Duplex DMX512 - Bidirectional use of the primary data link

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 signal packets as well as return data controlled by the use of Alternate START Code packets. There is no specific requirement that EF1 devices use NULL START Code packets for any of their functions – that requirement is defined by the specific EF protocol.

An EF1 system shall use polled responses on the primary data link, such that a receiving device can not transmit a packet unless instructed to by an appropriate Alternate START Code packet from the controller. All response packets shall begin with an Alternate START Code. Once configured, any functions using EF1 shall use collision free data protocols.

All packets in an EF1 system shall follow the normal timing and limits for a DMX512 packet. There may be additional timing limits imposed by the chosen protocol. Any additional timing limits are beyond the scope of this Standard.

B3.1.1 Bi-directional distribution amplifiers for EF1

Systems using EF1 topologies require bi-directional distribution/return data combiners. These combiners shall control the direction of data flow on the primary data link. The design of such a bi-directional distribution/return data combiner requires detailed understanding of the protocol timings. Therefore, the design of bi-directional distribution/return data combiners for EF1 without knowledge of the protocol to be used is meaningless. The requirements for such designs are beyond the scope of this Standard.

B3.1.2 Termination of EF1 system

Termination of EF1 primary data links shall conform to the electrical requirements clause 4.9. Additionally, EF1 systems shall terminate each end of the primary data link. Therefore, EF1 controllers and other transmitting devices shall provide a means to terminate the primary data link. Such terminations may also require specific line biasing or other methods to ensure integrity of returned data that are beyond the scope of this Standard.

B3.2 EF2 - Full duplex DMX512

The EF2 topology uses the second data link to provide a data path to return information from controlled devices. The data on second data link shall be unidirectional flowing from the controlled devices to the controller. This data link will operate in multi transmitter multipoint mode. In this mode the primary and secondary links comprise a full duplex data link.

Each responder shall have an EIA-485-A transmitter for the second data link with a driver enable control. The driver enable control shall be driven so that each responder can control the state of the return data link while transmitting its response bytes. Once configured only one responder shall be enabled at once and there shall be at least one bit time between one responder going inactive and the next going active. In general, the line driver can be enabled one bit time prior to transmission, and disabled one bit time after the last bit has been sent.

Once configured, any functions using EF2 shall use collision free data protocols. EF2 protocols shall also be structured to allow the use of data distribution amplifiers that meet the requirements of clause B3.2.2.

B3.2.1 Wiring of EF2 DMX512 ports

EF2 responders having two DMX512 ports for receive and transmit shall provide a direct passive link for Data 2- and Data 2+ between those ports (clause 4.8.2). The responder transmitter shall be connected to this passive link. Devices having three or more DMX512 ports shall wire the additional ports in a manner appropriate for the device's functionality. The manner in which these ports are wired shall be clearly detailed in the product's manual. Devices having three or more ports shall have two ports wired as a loop through with an attached responder transmitter.

B3.2.2 Bi-directional distribution amplifiers for EF2

Systems using EF2 topologies require bi-directional distribution amplifiers/return data combiners. The DMX512 primary data link may be split and separately buffered as in standard DMX512 buffers. Return data receivers shall be "wire-OR" connected within a unit. This combined received data signal is used to drive back to the return data monitor.

The bi-directional distribution amplifier shall not perform any processing on the data, since some response protocols depend upon the relationship with outbound DMX512 to synchronize the return data.

B3.2.3 Termination of EF2 data links

EF2 systems shall terminate the primary data link following the requirements of section 4.9. They shall also terminate the secondary data link to conform to the electrical requirements of 4.9 except that both ends of the secondary data link shall be terminated. Therefore, EF2 controllers and other transmitting devices shall provide a means to terminate the secondary data link. Such terminations may also require specific line biasing or other methods to ensure integrity of returned data that are beyond the scope of this Standard.

B3.3 EF3 Half duplex on the second data link

Systems that send data in both directions on the secondary data link are classified as EF3. These systems shall use the primary data link to send a standard unidirectional DMX512 signal.

B3.3.1 Bi-directional distribution amplifiers for EF3

Systems using EF3 topologies require bi-directional distribution/return data combiners. These combiners shall control the direction of data flow on the secondary data link. The design of such a bi-directional distribution/return data combiners requires detailed understanding of the protocol timings. Therefore, the design of bi-directional distribution/return data combiners for EF3 without knowledge of the protocol to be used is meaningless. The requirements for such designs are beyond the scope of this Standard.

B3.3.2 Termination of EF3 data links

EF3 systems shall terminate the primary data link following the requirements of clause 4.9. They shall also terminate the secondary data link to conform to the electrical requirements of 4.9 except that both ends of the secondary data link shall be terminated. Therefore, EF3 controllers and other transmitting devices shall provide a means to terminate the secondary data link. Such terminations may also require specific line biasing or other methods to ensure integrity of returned data that are beyond the scope of this Standard.

B3.4 EF4 Protocols that use the primary and the secondary data links in ways not covered above

Systems that use the primary and the secondary data links to send data in both directions are classified as EF4. Both data links shall comply with the requirements of clauses B3.1 (EF1) and fall within the scope of EIA-485-A except that the use of the secondary data link is optional.

B3.5 Additional electrical requirements

The details of a particular protocol using an EF topology may necessitate additional electrical requirements. These requirements are beyond the scope of this document.

-end of Annex B-

Annex C (Normative) - Higher Protection Levels – “DMX512-A Protected”

Some manufacturers feel it is prudent to employ higher levels of protection on their DMX512 ports than is specified in EIA-485-A. Many semiconductor manufacturers have recognized the need for higher protection and have produced “fault protected transceivers.” The tables below indicate the minimum requirements for a protected DMX512 port. If a manufacturer’s DMX512 port meets these requirements, the port can be declared as “DMX512-A Protected”.

Table C1 - Transmitter protection

Connection	Minimum Protection Limit		
Data 1- to Data Link Common or Data 1+ to Data Link Common	30 VAC / \pm 42 VDC		
Data 2- to Data Link Common or Data 2+ to Data Link Common	30 VAC / \pm 42 VDC		
Data 1- to Data 1+	30 VAC / \pm 42 VDC		
Data 2- to Data 2+	30 VAC / \pm 42 VDC		
	Ground Referenced	Isolated	Floating
Data Link Common to Chassis	< 0.2 ohms	N/A	N/A
Any other non Data Link Common Port Pin or contact to Chassis	30 VAC / \pm 42 VDC	N/A	N/A
Any Port Pin or Contact to Chassis	N/A	30 VAC / \pm 42 VDC	30 VAC / \pm 42 VDC

Table C2 - Receiver protection

Connection	Minimum Protection Limit		
Data 1- to Data Link Common or Data 1+ to Data Link Common	30 VAC / \pm 42 VDC		
Data 2- to Data Link Common or Data 2+ to Data Link Common	30 VAC / \pm 42 VDC		
Data 1- to Data 1+	30 VAC / \pm 42 VDC		
Data 2- to Data 2+	30 VAC / \pm 42 VDC		
	Non-Isolated	Isolated	Floating
Data Link Common to Chassis	-note 1-	N/A	N/A
Any other non Data Link Common Port Pin or contact to Chassis	30 VAC / \pm 42 VDC	N/A	N/A
Any Port Pin or Contact to Chassis	N/A	30 VAC / \pm 42 VDC 100Mohm at 50 VDC	30 VAC / \pm 42 VDC \geq 22Mohm at 50 VDC

Note 1: This cannot be characterized in terms of voltage. Clause A2 (Receiver Characteristics) allows manufacturers to fit a resistance between chassis and Data Link Common for the purpose of limiting the current in the shield due to small ground differentials. Any such resistance shall survive continuous connection to voltages within the EIA-485 Common Mode range of -7/+12 VDC.

-end of Annex C-

Annex D (Normative) - Reserved Alternate START Codes

D1 Reserved Alternate START Codes

Several Alternate START Codes are reserved for special purposes or for future development of the Standard. No equipment shall be manufactured that generates Alternate START Codes 92 - A9 (146 - 169 decimal) or AB - CD (171 - 205 decimal) until their use is defined by the Standard or by the E1 Accredited Standards Committee. Manufacturers shall not advertise or sell products or devices that use Alternate START Codes F0 - F7 (240 - 247 decimal).

Table D1 - Reserved Alternate START Codes

Alternate START Code		Purpose	Note
Hexadecimal	Decimal		
17	23	ASCII Text Packet	see Annex Clause D2 for implementation
55	85	Test Packet	see Annex Clause D3 for implementation
90	144	UTF-8 Text Packet	see Annex Clause D4 for implementation
91	145	91h followed by a 2 byte Manufacturer ID field is reserved for Manufacturer/Organization specific use, transmitted byte order is MSB, LSB. The next byte after the Manufacturers ID would normally be a manufacturer's sub-code	The Manufacturer ID serves as an identifier that the data following in that packet is proprietary to that entity and should be ignored by all others
92 - A9	146 - 169	possible future revisions of this Standard	Use not currently permitted
AB - CD	171 - 205	possible future revisions of this Standard	Use not currently permitted
CF	207	System Information Packet	see Annex clause D5 for implementation
F0 - F7	240 - 247	prototyping/experimental use while the manufacturer/organization is waiting for their registered Alternate START Code to be assigned	Not permitted in shipping product

D2 ASC text packet

Alternate START Code 17h (23 decimal) shall designate a special packet of between 3 and 512 data slots. The purpose of the ASCII text packet is to allow equipment to send diagnostic information coded per the American Standard Code for Information Interchange and formatted for display.

Slot allocation is as follows:

Slot 1: Page number of one of the possible 256 text pages.

Slot 2: Characters per Line. Indicates the number of characters per line that the transmitting device has used for the purposes of formatting the text. A slot value of zero indicates ignore this field.

Slots 3-512: Consecutive display characters in ASCII format. All characters are allowed and where a DMX512 text viewer is capable, it shall display the data using the ISO/IEC 646 standard character set. A slot value of zero shall terminate the ASCII string. Slots transmitted after this null terminator up to the reset sequence shall be ignored.

D3 ASC test packet

Alternate START Code 55h (85 decimal) shall designate a special test packet of 512 data slots, where all data slots carry the value 55h (85 decimal). Test packets shall be sent so that the time from the start of the Break until the stop bit of the 513th slot shall be no more than 25 milliseconds. When test packets are sent back to back, the Mark Before Break time shall be no more than 88 microseconds. The Break timing for test packets shall be greater than or equal to 88 microseconds, and less than or equal to 120 microseconds. The Mark After Break time shall be greater than or equal to 8 microseconds and less than or equal to 16 microseconds.

D4 UTF-8 text packet

Alternate START Code 90h (144 decimal) shall designate a special packet of between 3 and 512 data slots. The purpose of the UTF-8 Text Packet is to allow equipment to send diagnostic information coded per UTF-8 as described in Unicode 5.0 published by The Unicode Consortium and formatted for display. UTF-8 should only be used when the text packet cannot be expressed in ASCII per clause D2.

Slot allocation is as follows:

- Slot 1: Page number of one of the possible 256 text pages.
- Slot 2: Characters per Line. Indicates the number of characters per line that the transmitting device has used for the purposes of formatting the text. A slot value of zero indicates "Ignore this field."
- Slots 3 - 512: Consecutive display characters in UTF-8 format. All characters are allowed and where a DMX512 text viewer is capable, it shall display the data using the Unicode 5.0 character set. A slot value of zero shall terminate the UTF-8 text string. Slots transmitted after this null terminator up to the reset sequence shall be ignored.

D5 System Information Packet (SIP) Alternate START Code

Alternate START Code CFh (207 decimal) is reserved for a System Information Packet (SIP). The SIP includes a method of sending checksum data relating to the previous NULL START Code packet on the data link and other control information. No other packet shall be sent between the NULL START Code packet and the SIP that carries its checksum.

D5.1 Application

Manufacturers of control consoles are encouraged to transmit SIPs, either as a background to normal processing or, in conjunction with the special test packet, as part of their suite of system test functions. One of the current problems with testing of DMX512 installations is that it must be done with static test packets – certain modes of testers cannot be used while a console is actually running the show, as by definition the DMX512 packets are varying as each cue runs. The interleaving of SIP's would allow some degree of live testing, particularly if one of more test packets were also sent applicable to the functionality of the receiving device.

Note: For systems requiring a more reliable link, manufacturers would have the option of following every normal packet with a SIP packet, although it is recognized that this would degrade data throughput. It could be used with systems that send packets of fewer than 512 DMX512 data slots or refresh data at less than the maximum rate.

D5.2 SIP format

The SIP Packet Length is 24 data slots, in the format specified in Table D2. Receivers shall be required to accept SIPs of up to 255 data slots to allow for future expansion.

D5.3 SIP checksum pointer

Transmitting devices shall send a value of 24 in slot 1 that represents the length of SIP Packet in this version of the Standard. Receivers shall use the value received in slot 1 to establish the offset to the SIP checksum.

Table D2 - SIP Format

Slot	Definition	Refer to Clause
1	SIP Byte Count/SIP Checksum pointer (valid value is 24)	D5.3
2	Control Bit Field	D5.4
3	MSB of 16 bit additive Checksum of previous packet	D5.5
4	LSB of the 16 bit Checksum of previous packet	D5.5
5	SIP sequence number	D5.6
6	DMX512 universe number	D5.7
7	DMX512 processing level	D5.8
8	Version of Software sending this SIP	D5.9
9	Standard Packet Len MSB	
10	Standard Packet Len LSB	D5.10
11	Number of Packets transmitted by originating device since last SIP MSB	
12	Number of Packets transmitted by originating device since last SIP LSB	
13	Originating Device's Manufacturer ID MSB	D5.11
14	Originating Device's Manufacturer ID LSB	D5.12
15	Second Device's Manufacturer ID MSB	D5.13
16	Second Device's Manufacturer ID LSB	D5.13
17	3 rd Device's Manufacturer ID MSB	D5.13
18	3 rd Device's Manufacturer ID LSB	D5.13
19	4 th Device's Manufacturer ID MSB	D5.13
20	4 th Device's Manufacturer ID LSB	D5.13
21	5 th Device's Manufacturer ID MSB	D5.13
22	5 th Device's Manufacturer ID LSB	D5.13
23	<i>reserved for future use - transmit as 0</i>	
24 - (nn-1)	<i>reserved for future use</i>	D5.3
nn (max 255)	8-bit Additive Checksum of the SIP	D5.14

D5.4 Control bit field

d7	d6	d5	d4	d3	d2	d1	d0
reserved transmit as 0	reserved transmit as 0	reserved transmit as 0	reserved transmit as 0	reserved transmit as 0	reserved transmit as 0	previous packet bit set = 1	subsequent NULL packet hold control bit set=1

D5.4.1 Subsequent NULL packet hold control bit d0

Processing of the subsequent NULL packet hold Control Bit is optional. If implemented, when the subsequent NULL packet hold Control Bit is set (=1), the subsequent NULL START Code packet shall be held pending the reception of the next SIP for validation of the checksum. If a second NULL START Code packet is received without a preceding SIP, the receiver shall return to an immediate use mode.

D5.4.2 Previous packet bit d1

Set (= 1) if previous packet was an Alternate Start Code (ASC) packet.

D5.5 Checksums

16 bit ones complement additive checksum of all slots in the previous packet. The checksum includes the START Code.

D5.6 SIP Sequence number

A free running 8 bit counter identifying the SIP and incremented by a SIP generator by 01h on every subsequent SIP. This field may be checked to ensure that SIPs have not been missed.

D5.7 Originating universe

This slot indicates the (originating) DMX512 universe currently transmitted on this link. 00h is not used. Valid values 01h - FFh (1 decimal - 255 decimal).

D5.8 DMX512 processing level

This slot indicates the level of post controller processing. Originating devices shall always transmit a value of 00h in this field. Processing devices such as merge units or any that regenerate or provide a media conversion (e.g., Ethernet to DMX512) facility and do not explicitly block ASC test packets per clause D3 shall increment the value of this field by 01h. The content of this field indicates a level of process "hops" that data on the link has been subjected to relative to the originating transmitting device.

D5.9 Software version

00h	not implemented
01h - FFh	firmware version of last device

Note: This slot for use by the manufacturer and may not correlate with any formally published release identifier.

D5.10 Packet lengths

This declares the standard length of packets for START Code 00, normally transmitted on this link. Valid values are

0000h	packet length not declared
0001h - 0200h	designates value of fixed packet length
0201h - 7FFFh	are reserved
8000h	Dynamic Packet, length not declared
8001h - 8200h	length of last dynamic packet
8201h - FFFFh	are reserved

D5.11 Number of packets

A 16 bit count of the number of packets transmitted by the originating device since last SIP was transmitted. This count should not increment past FFFFh.

D5.12 Manufacturer ID

Manufacturer ID will be the same 16 bit assignment as used for the Manufacturer's ID field used with Alternate START Code 91h (see Annex E - clause E1).

an ID == 0000h indicates that Manufacturer is not declared.

an ID == FFFFh indicates that Manufacturer has applied for, but not been granted, and ID and that this transmission originates from a product under development.

D5.13 Packet history

DMX512 Processing devices and media converters that process SIPs shall be required to insert their own Manufacturer's ID into the SIP packet. An originating device shall always send its Manufacturer's ID in SIP slots 13 and 14, with 0000h in slots 15, 16; 17, 18; 19, 20 and 21, 22. Subsequent processing devices shall insert their manufacturer's ID into the slots as indicated by the DMX512 processing level slot. A

processing level of 01h corresponds to the second device, a processing level of 02h corresponds to the third device, and so on.

Note: This scheme allows for a packet processing history to be traced back through a complex installation of products.

D5.14 SIP Checksum

8 bit ones complement additive checksum of the SIP START Code (CFh) and all subsequent slots of SIP data.

-end of Annex D-

Annex E (Normative) - Alternate START Code, Manufacturer ID, and Enhanced Functionality Registration

E1 Alternate START Code Registration Policy: 1 - 255 decimal (01 - FF hexadecimal)

Slot 0 of a DMX512 packet is the START Code. The value of this slot identifies intended use of data in the rest of the packet. The Standard provides for a non NULL or "Alternate" START Code. Where it is required to send proprietary information over a DMX512 data link, a packet starting with a registered Alternate START Code shall be used.

E2 Authorized use

The E1 Accredited Standards Committee or any organization that it authorizes may use an Alternate START Code to provide further extensions to the DMX512 Standard.

E3 Reserved Alternate START Codes

Several Alternate START Codes are reserved for special purposes or for future development of the Standard. See Annex D.

E4 Requests for Registration of New START Codes

Any manufacturer or organizations involved in the use of DMX512 may request that a START Code or Manufacturer ID be registered for their use. Although not encouraged, an Alternate START Code and Manufacturer ID may be registered for proprietary use. Requests shall be forwarded to the Secretariat for the ANSI E1 Accredited Standards Committee – ESTA. ESTA will attempt to honor such reasonable requests as described below.

E4.1 Number of Alternate START Codes per entity

No more than one Alternate START Code may be registered to any one manufacturer/organization. Manufacturers and Organizations with Alternate START Codes registered prior to the publication of this Standard may request one additional Alternate START Code.

E4.2 Selection of the Alternate START Code value and Manufacturer ID

The assignment of any particular numeric START Code or Manufacturer ID value to any particular entity is solely at the discretion of Accredited Standards Committee E1. Assignment depends on the availability of unused and unreserved START Codes and Manufacturer IDs.

E5 Requirements for registration of an EF protocol

No EF protocol intended for use between multiple Manufacturer's will be registered as an EF protocol unless its basic structure is available to anyone by request and can be used freely by any manufacturer. The entire message structure for the protocol does not have to be made public, only the portions that are for public use. It is expected that some protocols may still have portions of the message structure that are reserved for proprietary use (i.e., vendor specific messages). Registration is at the discretion of Accredited Standards Committee E1.

E6 Documentation Register

E6.1 Documentation for use of Alternate START Codes

The manufacturer/organization requesting registration of an Alternate START Code shall provide a 2-line description of the purpose of the Alternate START Code. They shall list the minimum and maximum number of slots, including the START Code, in any proposed packet. Any provided description (subject to editing) shall be included in the Register. This is not required for functions associated with a manufacturer specific ID under Alternate START Code 91. It is recommended, but not required, for Alternate START Codes assigned prior to the adoption of this version of the Standard.

E6.2 Maintenance and Publication

The ANSI E1 Accredited Standards Committee through its secretariat (ESTA) shall maintain a Register of Alternate START Codes and Manufacturer IDs. ESTA will publish the Registry as needed.

E6.3 Supplemental documentation

If the manufacturer/organization wishes detailed documentation to be in the Public Domain, a note will be added to the Registry, but they will be responsible for such publication.

E7 Ownership

The DMX512 Standards are copyrighted. By registering a START Code or Manufacturer ID, no ownership rights are conferred to any third party. Alternate START Codes are registered to particular entities solely to allow for orderly management of the Standard. The registrant does not own the Alternate START Code or Manufacturer ID.

-end of Annex E-

- end of Standard -

ANSI E1.17-2006, Architecture for Control Networks - ACN Architecture

This document forms part of ANSI E1.17-2006, Entertainment Technology - Architecture for Control Networks, which was approved as an American National Standard by the ANSI Board of Standards Review on 2006 October 19.

Copyright © 2006 Accredited Standards Committee E1 and its secretariat, the Entertainment Services and Technology Association. All rights reserved.

ESTA TSP document reference number CP/2003-1007R4

Abstract

E1.17 "Architecture for Control Networks" (ACN) consists of a suite of protocols and languages which may be configured and combined with other standard protocols in a number of ways to form flexible networked control systems.

This document includes an informative introduction which describes the origins and aims of the protocol suite and gives examples of how the parts can be put together.

Following the introduction, the concepts and specifications which are needed within ACN to support the protocols of the ACN architecture but which are not a part of those individual protocols are defined. These include the common packet format used across several ACN protocols, definitions of components and use of addressing in ACN, the Root Layer protocol which interfaces to the underlying transport, the arrangement of protocol layering in ACN systems and use of interoperability profiles.

Table of Contents

1	Informative Introduction: Origins, Aims and Design Reasoning	4
1.1	Background	4
1.1.1	The Need for a New Control Protocol	4
1.1.1.1	Needs are greater	4
1.1.1.2	Capabilities are greater	5
1.1.1.3	Expectations are greater	5
1.1.2	Design Goals	5
1.2	ACN Design Reasoning	6
1.2.1	Getting and Setting Properties	6
1.2.2	Specific Functionality of Controllable Components	7
1.2.3	Components and Identity	7
1.2.4	Packing Multiple Messages and Multicasting	7
1.2.5	Reliability and Ordering	7
1.2.6	Discovery	8
1.2.7	Layering and Modularity	8
1.2.8	Efficiency Tradeoffs, Optimizations	8
1.2.8.1	Common Packet Format	8
1.2.8.2	Message Packing	8
1.2.8.3	Range Addressing	9
1.2.9	Root Layer Protocol	9
1.2.10	Transporting ACN protocols - use of TCP/IP	9
1.2.10.1	Flexibility in the choice of network media	9
1.2.10.2	What we do not get from TCP/IP	9
1.2.10.3	ACN Subset of TCP/IP	9
1.2.10.4	ACN With Other Transports	10
1.2.11	Operation of a Typical System	10
1.2.12	ACN in Other Environments and Uses - Interoperability Profiles	10
1.3	Other protocols within the ACN Architecture	11
1.3.1	Recursive use of PDU Format in Other Protocols	11
2	Normative Specification	11
2.1	ACN Components and Component Identifiers (CIDs)	11
2.1.1	Byte Ordering of CIDs	12

2.2	Protocol Data Units and Blocks - the Standard ACN Message Format	12
2.2.1	PDU	12
2.2.2	PDU Blocks	12
2.2.3	Nesting PDU blocks within PDUs	12
2.2.4	Use of PDUs and PDU blocks	13
2.2.5	Mixing and layering of PDUs and Protocols	13
2.2.6	Packets	14
2.2.7	PDU Block reception and order of PDU processing	14
2.3	Use of the Transport Protocol	14
2.3.1	Transport Protocol Addressing and ACN Filtering	14
2.3.2	The Root Layer Protocol	14
2.4	PDU Fields	15
2.4.1	Flags	15
2.4.2	Length	16
2.4.3	Vector	16
2.4.4	Header	16
2.4.4.1	Preservation of Protocol Layering	16
2.4.5	Data	17
2.4.6	Byte Ordering of PDU Fields	17
2.5	PDU Packing Rules	17
2.5.1	Suppression and Inheritance of Repeating Field Values	17
2.5.2	Inheritance at the Start of a PDU Block	18
2.5.3	Inheritance from Discarded PDUs	18
2.6	The ACN Root Layer Protocol	18
2.6.1	Root Layer Packet Format	19
2.6.1.1	Preamble	19
2.6.1.2	PDU Block	19
2.6.1.3	Postamble	20
2.6.2	Root Layer Protocol Operation	20
2.6.3	Root Layer Addressing	20
2.6.4	Byte Ordering in Root Layer Protocol	20
2.7	ACN PDU Structure Example	21
2.8	Protocol Identifiers	22
2.9	Interoperability Profiles	22
2.9.1	Naming of Interoperability Profiles	23
	Definitions	23
	References	23

1. Informative Introduction: Origins, Aims and Design Reasoning

Non-normative

The Architecture for Control Networks arose from a need within the entertainment technology industry in general and particularly the lighting industry for a common interoperable protocol or protocols for control of equipment which would take advantage of the new technologies which were gaining dominance and offering cheap high speed communications in many areas.

This section describes the background from which ACN was developed, the design reasoning and processes which have gone into generating ACN and the way a typical implementation fits together. It is informative only, but is recommended reading as starting point for those new to ACN.

In designing ACN, great reliance has been placed on a wide variety of existing technologies which are constantly changing and progressing. The protocols of ACN have well defined functionality and clear boundaries and layering. It follows that all these pieces can and will be put together in different ways to suit different applications. The things which tie all these pieces together are interoperability profiles which specify which pieces must be used and what their operational parameters must be to achieve interoperability over a particular application domain. They may also specify what form of bridging or translation is required to connect them to networks using different profiles.

Thus the ACN architecture consists of a number of separate specifications for protocol formats, languages etc., together with interoperability profiles which specify where and how those specifications are to be used and call up external specifications as necessary. The driving force for ACN has been control of entertainment technology equipment across mainstream networks such as Ethernet and the initial interoperability profiles and specifications reflect this. However, other applications can be developed within this framework.

1.1. Background

The following summarizes the situation at the beginning of the ACN project:

1.1.1. The Need for a New Control Protocol

The lighting industry has been served well since the late 1980s by the USITT DMX512 data link standard for interconnecting controllers with dimmers [DMX512]. Other entertainment technologies have other data transport standards both proprietary and open e.g. MIDI Show control, MIDI Machine control, PA-422 and a plethora of proprietary systems in the audio industry. However, a lot has changed since these were introduced and in particular the explosive growth of sophisticated networking since the late 1990s.

1.1.1.1. Needs are greater

Typical lighting systems integrate the control of a large and growing variety of automated fixtures, color scrollers, and other devices where previously there were just dimmers. Furthermore, the lighting control system is expected to integrate with systems controlling sound, automation, environmental equipment, pyrotechnics and a host of other things. With the rise of themed environments of various kinds, the line

between performance control and architectural and environmental control has become blurred, if not erased.

1.1.1.2. Capabilities are greater

Along with the phenomenal growth of business LANs and the global Internet have come a variety of powerful networking technologies and general-purpose protocols available off-the-shelf. Meanwhile, control consoles and processor units are using embedded computers with mainstream operating systems and hardware, capable of supporting modern networks. The technology is ready for the application of advanced computer networks to entertainment technology control.

1.1.1.3. Expectations are greater

Professionals in entertainment technology use computers, LANs, and the Internet, both at work and at home. They know what computers and computer networks are capable of, and they expect as much from their control systems. Standardization has been a key force in mainstream computing, allowing an enormous variety of equipment from thousands of different manufacturers to “plug-and-play”. This is a model of what is possible, and users of lighting equipment wonder, quite rightly, why a similar model cannot be realized in lighting control. The ESTA Architecture for Control Networks (E1.17) [ACN] is an effort to address these needs, capabilities, and expectations.

To ensure continued success and growth, the industry needs flexible, scalable, and powerful industry-standard mechanisms for discovery, configuration, and control. The challenge that ACN meets is to provide that power and flexibility while still enabling lightweight devices to participate.

1.1.2. Design Goals

To guide the development of ACN a set of design goals for an control network were enumerated at the start of the project. These design goals serve as a set of minimum requirements against which ACN can be judged.

1. **Interoperability across manufacturers.** The protocol should enable various participating manufacturers' equipment to communicate usefully. e.g. to let one manufacturer's console control another manufacturer's dimmers or moving lights.
2. **Many sources, many sinks.** The protocol should provide for multiple sources of control data on the same network, as well as multiple consumers of that data. Sources and sinks should not be equated with controllers and controlled devices - data will flow both ways.
3. **One “wire”, but multiple, independent uses.** The protocol should allow a single network to support multiple independent uses. For example, in large, complex installations it will be desirable to dynamically configure sub-venues as independent universes of control, with independent addressing, etc. Another example: while integration of audio and lighting control may be possible, it is not always desirable and the two systems would frequently be operated independently.
4. **A mainstream protocol.** The protocol should be capable of traveling over mainstream network protocols. It should not attempt to reinvent the functionality of lower-layer network protocols.
5. **Maximum opportunity to use off-the-shelf technology.** The protocol should be designed to exploit the enormous variety of off-the-shelf networking hardware and software (routers, switches, protocol

stacks, diagnostic tools, etc.) available from third parties. Moreover, it should be designed so that it can continue to “ride the wave” of the rapidly evolving state of the art in networking in the future.

6. **Support for manufacturer-specific uses.** Only a subset of the control and data requirements of a modern entertainment technology installation will be standardized industry-wide. The protocol should support manufacturer-specific extensions in an elegant way. The subset that is standardized should not be a conceptual orphan, but should fit in naturally with the protocol as a whole.
7. **Scalability.** The protocol should adequately address the needs of the full range of size and complexity of applications. It should be simple enough for use on the smallest systems, and also scale up to networks controlling large Broadway Musicals, hotel complexes, theme parks, etc.
8. **Extensibility.** The protocol should be designed to be easily and cleanly extended over time to handle future requirements as they arise. The protocol must be “built to last” and “future proof”.
9. **Ease of configuration.** The protocol should provide a means for the network to be easily configured and managed. Devices should dynamically discover one another and one another's capabilities without user intervention: “plug-and-play”.
- 10 **Efficient and predictable use of available bandwidth.** The protocol should be reasonably frugal with network bandwidth. This requirement follows from design goals 2, 3, 7, 8, above. For example, rather than continually updating the level of a device it should be possible to send only changes in output levels. Moreover, network bandwidth usage should be predictable, so designers of new systems can correctly specify network requirements.
- 11 **Flexibility and control with respect to sub-networks and routing.** To achieve scalability and efficient use of available bandwidth, it is essential that the protocol does not foreclose the normal range of options used to structure network traffic. E.g., network designers must be free to use sub-net addressing and routers in ways that reflect the actual network structure. Sub-net addressing provides a means for limiting network traffic by isolating sub-nets in complex networks. For example, the protocol should not use sub-net addressing in ways that force all devices of a given type to reside in the same sub-net.
- 12 **Fault tolerance.** The kinds of applications in which ACN will be utilized cannot tolerate frequent network failures. Nor can end-users be expected to have a high level of networking expertise. Ideally, network failures will be minimized and the network will recover gracefully and with a minimum of user intervention when failures do occur.

1.2. ACN Design Reasoning

This section gives a very brief overview of the design reasoning behind ACN and introduces the main elements of the suite. Much of this section describes the particular use of ACN protocols over TCP/IP for device control using Device Management Protocol. It is important to bear in mind that while this usage drove much of the development, many other configurations are possible and have been considered.

1.2.1. Getting and Setting Properties

Controlling ACN devices is achieved by representing the functions of that device as a set of variables or “properties”. The device is monitored by “getting” the values of these properties and controlled by

“setting” them. In ACN this most fundamental function is performed by Device Management Protocol (DMP) [DMP] whose most basic messages are `Get_property` and `Set_property`. DMP also defines an addressing scheme for variables within a device. DMP does not itself apply any meaning to those properties - a dimmer level is no different in DMP from a telescope azimuth.

1.2.2. Specific Functionality of Controllable Components

DMP provides a generalized method for Getting and Setting values of variables in components. Without knowing what those variables represent, a controller can achieve nothing. It would be possible to use a scheme where properties for specific functions are always at the same address, but this is very unwieldy and fails on goals 7 and 8. In ACN the mapping between properties and specific device function is provided by a separate device description. Device Description Language (DDL) [DDL] defines the format and language for these descriptions in a way that allows a controller to find the functionality of each property. The DDL model not only allows simple controllers to seek out the properties for functions it knows about, such as intensity or gobo number, but also allows smarter controllers to be designed which can work out how to deal with functions which they have never met before.

1.2.3. Components and Identity

In a complex networking environment, there is not a one-to-one correspondence between network interfaces and the devices or processes which are sending or receiving the data e.g. one single device may have multiple interfaces, while a computer with just a single interface may be running two or more completely independent ACN applications. So some mechanism is needed to identify and address the transmitter or receiver of data.

There is also a need for persistent identity of devices independent of network changes. For example, a motion controller needs to relate a particular winch (among many identical ones) to particular actions which the controller has stored for it. In many systems that same winch may be assigned a different network address the next time the system is powered up.

In ACN each distinct endpoint transmitting and receiving ACN data is called a *Component* and all ACN communications take place between *components*. Each ACN component has a *component identifier* or *CID* which is unique not just within the system but across the whole world and does not change with time.

A device such as a dimmer or audio amplifier will typically be a single ACN component, but a large control console or a general purpose computer may contain multiple components.

1.2.4. Packing Multiple Messages and Multicasting

`Get/Set_property` messages are typically very short, but transports like Ethernet and TCP/IP are most efficient when transferring large blocks of data and become inefficient when packets become very small. To combat this mismatch, ACN allows large numbers of short messages to be packed into a single *packet*. A packet containing multiple messages is then sent to all the devices concerned in a single transmission and those devices extract their own messages. Sending a single packet to a selected group of devices is called “multicasting”.

1.2.5. Reliability and Ordering

If a Set_property message to a device goes missing or arrives out of order, that device may be in an incorrect state. A reliable transport which ensures that messages do arrive and in the correct order, or if there is a failure, the controller knows about it can relieve controllers of a large amount of effort in checking that messages have arrived and that the state of the device is as expected. However, in a multicast protocol a straightforward acknowledgement of every message cannot work since the outgoing packet may be addressed to hundreds of devices and if they all acknowledge, the network becomes flooded – so a more sophisticated and efficient reliable multicast technique is required. In ACN this is achieved by Session Data Transport (SDT) [SDT] which can provide reliable, *ordered* delivery of multicast messages.

1.2.6. Discovery

Ease of configuration (goal 9), means that human operators should not have to tell the controller which devices are present and how to control them. The process of automatically finding and organizing devices and working out how to control them is called discovery. In ACN it takes several stages. 1. the controller has to find which ACN devices are on the network. 2. it has to find out what sort of devices they are. 3. it has to find the structure of the devices so that it can control them if required.

In IP networks Service Location Protocol (SLP) [SLPv2] is used for the first stage and the initial part of 2 (discovery of the root device type). Further discovery of device type and stage 3 is done by examination of the DDL description of particular devices.

1.2.7. Layering and Modularity

In accordance with design goals 3, 6 & 8 there is a benefit in separating the reliable multicast technique from the Get/Set_property protocol. This is why SDT, DMP and DDL are separate. It allows alternative protocols for other purposes (e.g. proprietary protocols, show control, time code, file transfer, content streaming...) to use the reliability and group management provided by SDT. It allows DMP to be used over other transports if the SDT model is not appropriate. Finally and most important, dividing the protocol into layers simplifies it because each layer is only concerned with a well defined part of the problem.

1.2.8. Efficiency Tradeoffs, Optimizations

1.2.8.1. Common Packet Format

In developing SDT and DMP it was recognized that the messages in both protocols have many similarities and that by using a common message format across all the ACN protocols, the code to recognize and decode messages can be shared across different message types. This is why ACN defines a common message format the *Protocol Data Unit* or *PDU*.

1.2.8.2. Message Packing

A network protocol can be optimized for (among other things) packet size, processing speed or processing code size. Any such optimization involves trade-offs. Goal 10 of ACN is efficient use of bandwidth. The ACN PDU format provides a number of optimizations to allow the number of messages packed into a single *packet* to be maximized. This involves a small increase in processing complexity but analysis has shown packing improvements of a factor of four or more in some common circumstances. Furthermore the small increase in processing complexity is offset by the efficiency gain and

simplicity provided by using a common PDU format across all PDUs and layers.

1.2.8.3. Range Addressing

At the DMP layer, the addressing scheme and use of range addressing allow very large numbers of property values to be transferred with minimal overhead for transmitting their addresses. This adds to the bandwidth efficiency of DMP based systems.

1.2.9. Root Layer Protocol

Because of the way the common packet format is used, the modularity of the protocols and the desire to allow a common entry point in the stack for all ACN protocols, the Root Layer protocol is defined. The Root Layer Protocol specifies how PDUs from different ACN protocols may be combined into packets for transmission via the transport, and how they are separated and passed for processing on receipt.

1.2.10. Transporting ACN protocols - use of TCP/IP

Careful separation of the ACN modules allows it to operate over many transports. However, initial LAN applications of ACN are expected to use TCP/IP and this provides a good model for future use with other transports.

The near universal acceptance of the TCP/IP family of protocols means that there exist an enormous number of commercially available hardware and software products for the implementation of such networks. Moreover, the continuing commercial momentum behind TCP/IP means that as a platform it will continue to evolve and be supported for the foreseeable future. These kinds of considerations make a compelling case for using the TCP/IP protocols, or a suitable subset of them, as the lower-level protocols within ACN systems.

1.2.10.1. Flexibility in the choice of network media

TCP/IP provides an abstraction of a network so that application programs that communicate via TCP/IP generally do not care, or even have to know, what the underlying network technology is. Technologies commonly used at present for carrying TCP/IP include IEEE 802.3 (wired Ethernet) at a variety of data rates, IEEE 802.11 (wireless Ethernet or “WiFi”), IEEE 1394 (Firewire), modems and other serial links and high speed links for “backbone” distribution such as ATM, SONET and FDDI. As new technologies emerge and TCP/IP is adapted to them, so applications that communicate via TCP/IP will be portable to these new network technologies. Thus when run on TCP/IP, ACN protocols are not Ethernet protocols, but enjoy the full benefits of infrastructure and tools created for TCP/IP networks. Because of its modularity, ACN will not be made obsolete by the appearance of new network technologies. Instead, applications that employ ACN will be positioned to “ride the wave” of advances in network technology.

1.2.10.2. What we do not get from TCP/IP

There are currently no reliable multicast protocols that suit the needs of the ACN architecture, therefore we have developed SDT for reliable multicast and session management in ACN.

1.2.10.3. ACN Subset of TCP/IP

ACN only uses a subset of the large TCP/IP suite and every attempt is made to keep this subset as small and simple as possible. In particular operation of SDT over UDP [UDP] is possible without any use of

the complex TCP protocol itself (Confusingly, TCP is just one optional part of the TCP/IP suite although it has given its name to the whole suite).

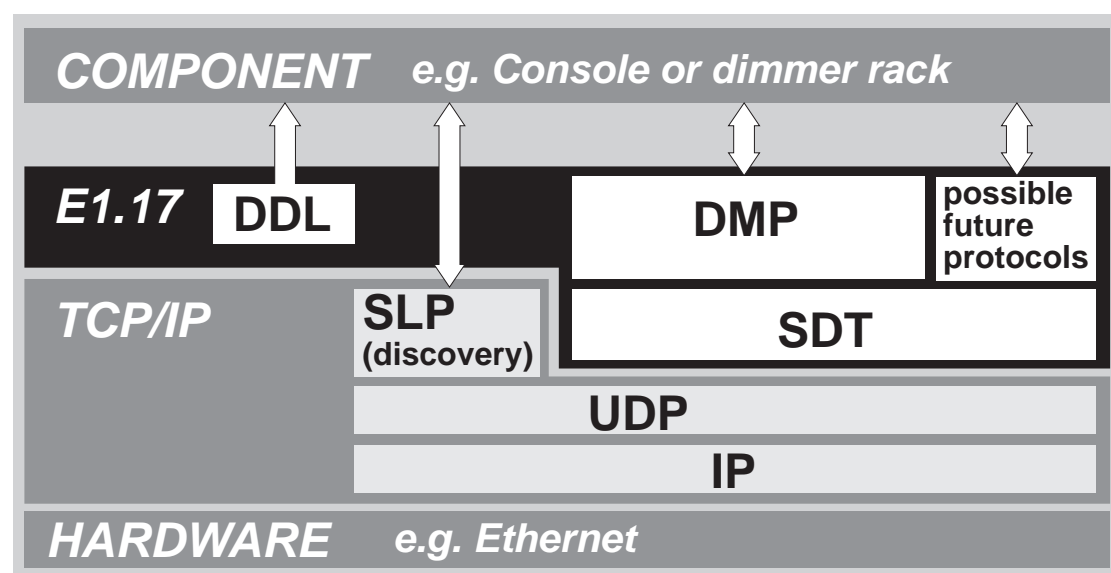
1.2.10.4. ACN With Other Transports

In the interests of good modular protocol design, the interface between ACN protocols and TCP/IP is clean and well defined and it is quite possible to separate ACN from TCP/IP where it is required to operate over alternative transports. Maintaining this division will also assist in addressing any issues with migration from version 4 to version 6 and above of the TCP/IP suite.

1.2.11. Operation of a Typical System

In an ACN system for control of devices over Ethernet, a controller will send Get/Set_property messages to devices as defined by DMP. These messages are transmitted using SDT which provides reliability, online status and management of groups of devices. All DMP and SDT messages are packed in individual PDUs using the common PDU format. These are then transmitted using UDP over Ethernet. New devices coming online are initially discovered using SLP and their specific functionality is then analyzed and configured using DDL.

Figure 1. ACN Modules



Note: It is important to bear in mind that while this usage drove much of the development, many other configurations are possible and have been considered.

1.2.12. ACN in Other Environments and Uses - Interoperability Profiles

The protocols resulting from the ACN project have followed a very modular design process and every attempt has been made to make modules independent of one another - this allows modules to be added, omitted or recombined as required. Furthermore within modules many parameters are not fixed but are necessarily dependent for optimal operation on the settings of other modules or on environmental factors

such as network topology, latency or reliability requirements.

Because of this the core protocol suite does not define fully how interoperability is to be achieved in any particular application area. This problem is addressed by producing interoperability profiles which define what combinations of modules and parameters must be used to gain interoperable operation in a particular environment (e.g. for a Local Area Network operating over UDP/IP on Ethernet). Interoperability profiles will also spell out what is needed to link pieces of equipment which conform to other profiles - this might be as simple as bridging one physical network type to another (e.g. Ethernet to IEEE 1394 - “Firewire”) or may require more complex processing (e.g. when building a proxy router between an Ethernet/IP network and an RS485 serial implementation).

Interoperability profiles are separate from the protocol suite itself but are required to define an implementation. The section “ACN Operation In a Nutshell” above actually describes an interoperability profile for control of devices using DMP over SDT over UDP over IPv4 over Ethernet - probably specifically CAT-5 Twisted pair topology.

1.3. Other protocols within the ACN Architecture

It is not expected that ACN will remain a static standard. The current protocols may be extended or refined and new protocols may be added within the overall architecture.

1.3.1. Recursive use of PDU Format in Other Protocols

The payload of root layer PDUs are passed on to higher layer protocols and it is entirely up to those protocols to specify the format and use of the data. However, within ACN as currently specified, there is just one higher layer protocol defined which interfaces directly to the root layer: Session Data Transport [SDT] is defined within the ACN suite for providing reliable ordered delivery of messages in a multicast context. It not only receives its data from root layer PDUs, but uses the same PDU format internally - with multiple nesting levels - for organizing its messages. In SDT the vector field within the PDU is used to specify the message type or command code.

Device Management Protocol [DMP] is the protocol specified within ACN to dynamically control things such as lights, mechanical equipment, audio devices etc. DMP is transported by SDT and like SDT, DMP also uses the PDU format for data packing.

This recursive use of a common data packing format allows efficient implementation of ACN systems in small devices because, a single body of code can be optimized and debugged and then reapplied at all layers.

2. Normative Specification

2.1. ACN Components and Component Identifiers (CIDs)

There is not a one-to-one relationship between data sources/sinks and network interfaces. Firstly a single piece of equipment quite commonly has multiple network interfaces either because it supports more than one network medium (e.g. Ethernet and a modem connection) or because it is connected to more than one network segment; and secondly a single computer may be running two or more independent programs using ACN, which have no specific knowledge of each other and yet are required to share one

network interface.

An ACN *Component* is a distinct endpoint transmitting and receiving ACN data and all ACN communications take place between components.

Every ACN Component shall be identified by a *Component identifier* or *CID*. A CID shall be a UUID [UUID] which is a 128-bit number which is unique across space and time. Each piece of equipment should maintain the same CID for its entire lifetime (e.g. by storing it in read-only memory). This means that a particular component on the network can be identified as the same thing from day to day despite network interruptions, power down and so on. However, in some systems there may be situations in which volatile components are dynamically created “on the fly” and in these cases the controlling process can generate CIDs as required. The choice of UUIDs for CIDs allows them to be generated as required without reference to any registration process or authority.

2.1.1. Byte Ordering of CIDs

When CIDs are transmitted within ACN protocols (e.g. SDT), they shall be ordered in network byte order as specified in section Section 2.4.6, “Byte Ordering of PDU Fields” and section 4.1.2 of [UUID].

2.2. Protocol Data Units and Blocks - the Standard ACN Message Format

2.2.1. PDUs

Protocols within the ACN suite use a common format which generalizes the structure of a *message*, command or data item as a *Protocol Data Unit (PDU)*. The information within a PDU and its format are defined below.

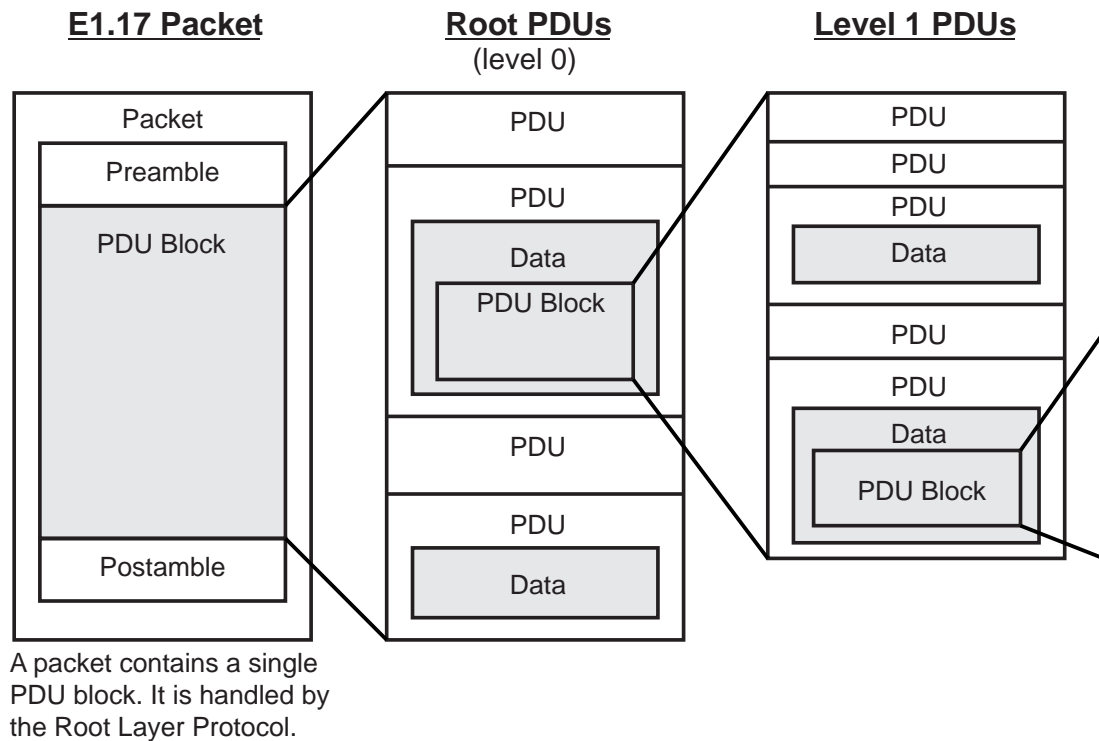
2.2.2. PDU Blocks

Multiple PDUs may be packed together contiguously in a single data block which is called a *PDU block*.

Within ACN the term *PDU block* shall mean a contiguous set of zero or more PDUs with no intervening non-PDU data, which are packed together according to the packing rules defined below. In order to process a PDU block, its size or the number of PDUs within it must be provided by the context in which it occurs.

2.2.3. Nesting PDU blocks within PDUs

Each PDU in a PDU block includes a data section (see below) and this may itself contain one or more PDU blocks - depending on the specification of individual protocols.

Figure 2. Example of PDU Nesting in an ACN Packet**Note**

A more concrete example of PDU nesting and usage is given in Section 2.7, “ACN PDU Structure Example”

2.2.4. Use of PDUs and PDU blocks

PDUs and PDU blocks are the basic building block used to make ACN *packets*.

ACN protocols define messages (or commands) and the rules for their combination or contents and these rules dictate how the PDU structure is used and combined to do the work - the structure is generic but the rules and usage are specific to individual protocols.

2.2.5. Mixing and layering of PDUs and Protocols

PDUs of multiple protocols may appear within a single PDU block subject to any restrictions specified with individual protocols. Protocols may also use PDU nesting for packing data within the single protocol (both SDT and DMP do this).

An ACN protocol may specify within which other PDUs its defined messages (themselves PDUs) may be nested. A protocol may also specify exactly what messages may be embedded within its message PDU blocks. This allows the protocol to specify more exactly how its part of ACN packets are structured.

Protocols carried within the ACN architecture may or may not use the ACN Packet Format for their data (e.g. to encapsulate individual commands). Because the Root Layer Protocol uses PDUs to pack data into a packet it follows that all protocol data within an ACN packet must be enclosed (nested) directly or indirectly within a PDU at the root layer and may additionally be nested at higher layers.

2.2.6. Packets

Within ACN, the term *packet* shall mean a datagram which is carried by an underlying transport protocol and is not wrapped within any outer layer ACN PDU. Packets are transmitted and received by the Root Layer Protocol as defined below.

2.2.7. PDU Block reception and order of PDU processing

On receipt of a PDU block, all PDUs within that block shall be examined and processed strictly in the order in which they occur in the block. In this context, processing refers to unpacking and computing field values and following all E1.17 rules for handling PDU Blocks and PDUs as specified below (see Section 2.4, “PDU Fields” and Section 2.5, “PDU Packing Rules”). ACN protocols shall also carry out further protocol specific processing of PDUs strictly in order in which they occur in the block unless otherwise specified.

2.3. Use of the Transport Protocol

Communication in an ACN system takes the form of packets traveling from one component to one or more other components. These ACN packets must be sent using some underlying protocol - the transport protocol (e.g. UDP). Passing the ACN packet to the transport protocol does this. In addition to the Packet, the Transport protocol will require information such as the following:

- The transport protocol address of the receiver of the ACN packet;
- The transport protocol address of the sender of the ACN packet;
- The length of the ACN packet;
- The data that makes up the packet.

The addresses used by the transport protocol are dependent on that protocol. See annexes and EPIs for details of using ACN on specific protocols.

2.3.1. Transport Protocol Addressing and ACN Filtering

Because underlying transport protocols have no concept of ACN components, an association is required between an ACN component and a transport protocol address. This association does not guarantee that the transport protocol delivers the packet exclusively to a single component and indeed some parts of ACN rely on multicasting for delivery to multiple destinations at once. Thus an ACN implementation has to filter PDUs received and each component can expect to reject many PDUs which are not intended for it. However, ACN protocols should use the addressing mechanisms of the transport protocol to minimize the set of recipients wherever possible. This process is described more fully in [SDT].

2.3.2. The Root Layer Protocol

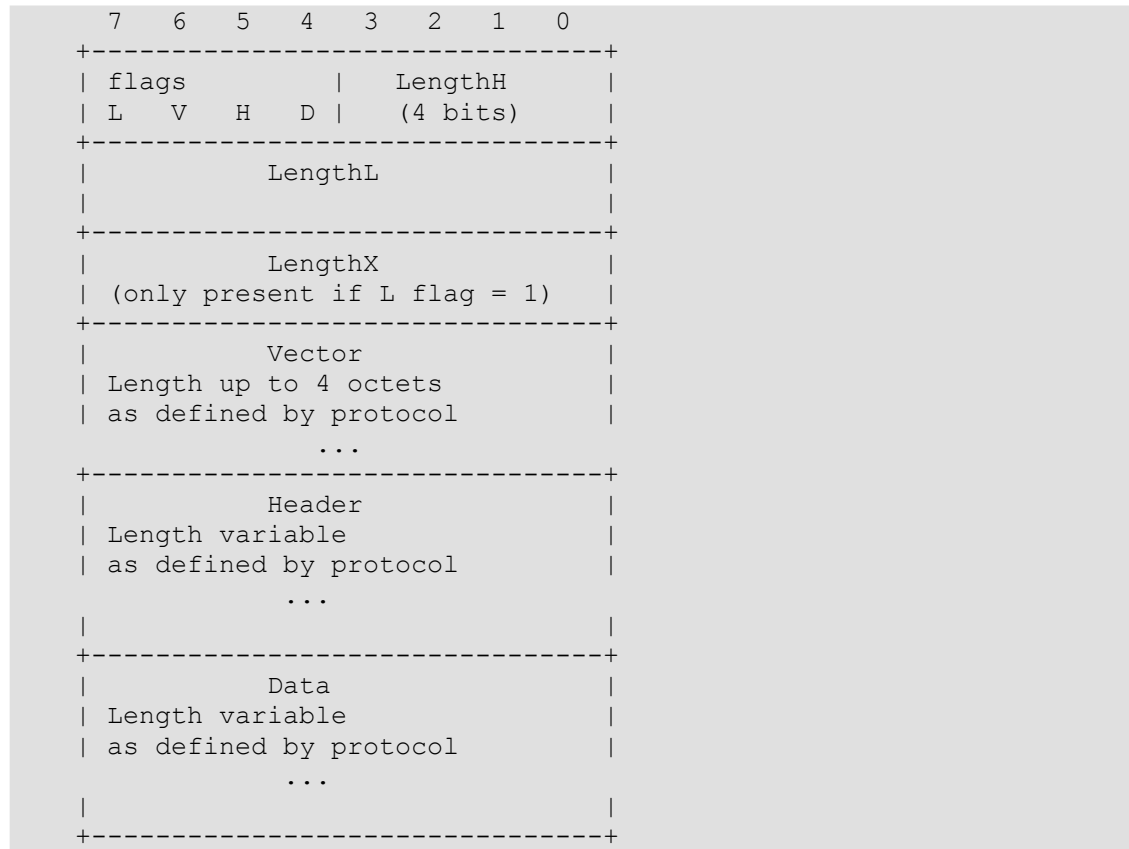
The lowest ACN layer above the underlying transport (e.g. the contents of a UDP packet) is handled by the ACN Root Layer Protocol. The Root Layer protocol may be tailored to fit higher ACN protocols onto each individual transport.

2.4. PDU Fields

Each PDU in a PDU block shall consist of 5 fields: Flags, Length, Vector, Header and Data. With the exception of Flags and Length which are packed together, all fields shall occupy an integer number of octets (which may be zero).

A PDU is laid out in octets as follows:

Figure 3. Layout of PDU



2.4.1. Flags

Flags is a 4-bit field containing flags L, V, H and D which declare how the PDU is packed as defined below.

2.4.2. Length

Length shall specify the length in octets of the entire PDU including Flags, Length, Vector, Header and Data fields as it occurs in the packet after all packing has been applied. i.e. Length is the number of octets from the start of this PDU to the start of the next in the block (or to the end of the block). This allows rapid traversal of PDUs in a block.

If the PDU length is 4095 octets or less, the L flag shall be set to zero and the PDU length shall be a 12-bit quantity with bits 11..8 (most significant bits) contained in PDU octet 0 bits 3..0, and length bits 7..0 contained in PDU octet 1. (fields LengthH and LengthL respectively in the diagram above).

If the PDU length is 4096 octets or greater (up to 1048575 octets) the L flag shall be set and the PDU length shall be a 20-bit value, with bits 19..16 (most significant bits) contained in PDU octet 0 bits 3..0, length bits 15..8 contained in PDU octet 1 and length bits 7..0 contained in PDU octet 2. (fields LengthH, LengthL and LengthX respectively in the diagram above).

Note that all allowable PDU lengths have a required encoding – there is never a choice of encoding. A consequence of this is that in an implementation where other restrictions (such as MTU) dictate that length must always be less than 4096 octets, the longer 20-bit length can never occur and length will always occupy 12 bits with L cleared.

2.4.3. Vector

Vector is a short, fixed length field indicating to the receiver, how and where to pass this PDU on for the next level of processing – conceptually it points to a handler for the payload of the PDU. Depending on the rules of individual protocols it may for example identify the message type of the PDU (command code), the protocol decoder for the PDU (protocol identifier) or the recipient data stream (demultiplex channel).

For example in the case of the Root Layer Protocol, Vector identifies an ACN protocol such as SDT and the entire PDU is passed to the handler for that protocol. At SDT's base layer the Vector field holds the message identifier (command code) while within an SDT wrapper, vector indicates the recipient component (by memberID).

The size of Vector shall be no greater than four octets. The size of Vector shall be the same for all PDUs in any PDU block. The specific size of Vector within a PDU block shall be a fixed by the protocol rules defining that PDU block.

2.4.4. Header

Header supplies supplementary information on the content of the PDU. The size, use and format of the Header field depends on individual protocol definitions. The Header field may be of zero size.

To ensure that PDUs may be unpacked efficiently, the protocol rules defining and PDU block shall either specify a constant header size for all PDUs in the block or, if the header is of variable size, shall specify an encoding which allows its size to be readily calculated by an unpacking routine without reference to the content of any other PDU field (for example, by placing the header length in the first octet of the header field). This is necessary so that the division between the header and data fields can always be found by an unpacking routine even if the vector or other parts of the header field are not recognized.

2.4.4.1. Preservation of Protocol Layering

Where layered protocols each use the PDU format, PDU nesting levels need not correspond directly to protocol layers. However, where nesting levels cross protocol layers, (i.e. in any PDU whose data is to be passed to another protocol for processing), “opaque” data belonging to the identified protocol shall be carried entirely in the Data field and Header shall not be used to carry client protocol data. This rule ensures clean separation of protocol layers.

2.4.5. Data

Data holds the content of the PDU and its interpretation is defined by the protocol, message type or other handler identified by the Vector field. The Data field may be of zero size. Because the length of the entire PDU is known and the size of all other fields is known, the length of the data field can always be calculated.

The use of separate Header and Data fields gives more scope for protocols to optimize away information which is repetitive or redundant from one PDU to the next. (see Section 2.5.1, “Suppression and Inheritance of Repeating Field Values”).

2.4.6. Byte Ordering of PDU Fields

Length and Vector fields within the ACN PDU format shall always occur in network byte order (big endian).

Byte ordering of Header and Data fields is dependent on individual protocol specifications.

Byte order for all protocols defined as part of the ACN suite shall be network byte order (big endian) unless explicitly stated otherwise. Protocols defined as part of the ACN suite should not use any other byte order for any field unless exceptional reasons prevail.

2.5. PDU Packing Rules

Analysis of the type of data which ACN is dealing with shows repetitive structures at many levels in which certain parts of the information do not change from one item to another. The ACN PDU packing aims to allow efficient compression of the data by identifying parts which repeat from one PDU to the next and allowing all but the first occurrence to be omitted from the packet. This is applied to the Vector, Header and Data fields.

2.5.1. Suppression and Inheritance of Repeating Field Values

The three flags V, H and D represent the state of each of the three fields Vector, Header and Data in the PDU respectively.

When constructing a PDU block, any of the three fields, Vector, Header and Data may be omitted if its value is identical to the same field in the preceding PDU in the block. If the value of a field differs from the value of the same field in the preceding PDU in the block then that field shall be present in the PDU.

If a field is omitted (because its value was identical to that in the preceding PDU), then the corresponding flag shall be set to zero. If the field is present, then the corresponding flag shall be set to one.

On unpacking a PDU block, if the PDU flag corresponding to a field is “1”, then the value to be used for the field shall be present in the PDU. If the flag is “0”, then no value shall be present in the PDU and an “inherited” value for the same field from the previous PDU shall be used. In either case the value of the field must be kept available in case it is inherited by the next PDU in the same block.

Note that for Header and Data there is a difference between an inherited value ($H = 0$ or $D = 0$) and a non-existent value which is indicated by $H = 1$ or $V = 1$ and a length of zero. If two or more successive PDUs contain a field of zero length, the second and subsequent PDUs may specify the value explicitly (H or $V = 1$) or by inheritance with no change to the resulting PDU length.

2.5.2. Inheritance at the Start of a PDU Block

The previous value of a PDU field shall not be inherited over from one PDU block to the next. It follows that at the start of a PDU block before unpacking the first PDU, no previous value is available for any field.

Any PDU containing one or more fields whose value is not available for any reason shall be discarded and the next PDU in the block (if any) processed in turn.

Any field value which is inherited from an unavailable value shall itself defined to be unavailable.

2.5.3. Inheritance from Discarded PDUs

It is legal for a PDU to inherit field values from a preceding PDU which itself has been discarded. This could have occurred, for example, because the vector field is unrecognized or because one of its field values was unavailable. This implies that a receiver implementation must retain and be able to identify any available field values in a preceding PDU even when that PDU was discarded.

2.6. The ACN Root Layer Protocol

The purpose of the Root Layer Protocol in ACN is to join base-level protocols onto the transport layer. To do this it must fulfill two functions:

Multiplexing/Demultiplexing

Multiple protocols may exist at the lowest(transport) layer of the ACN architecture. For example, in the future both SDT and XYZP (a purely hypothetical protocol) may be used concurrently. In this situation the Root Layer protocol packs PDUs from both of these protocols into packets for transmission (multiplexing) and unpacks them and passes them to the appropriate receiving code on reception (demultiplexing). This allows multiple protocols to coexist on a common underlying transport.

Additional Information

Different underlying transports may vary wildly in the services they provide. Additional information may be necessary or desirable for successful processing of ACN packets. The Root Layer protocol provides the preamble and postamble fields as places to add this information. Typical information in these fields include packet length, error checking (checksum, CRC etc.), packet identification (e.g., on IPv4 Networks the preamble contains a text string to help identify ACN packets for network diagnostics, see [RLP-UDP]).

The term “Root PDU” or “Root Layer PDU” shall mean a PDU which is extracted from the packet by the Root Layer Protocol and so is not enclosed in any outer PDU.

The Root Layer Protocol may be tailored to fit the ACN Architecture onto a wide variety of different transports which may have different requirements. Therefore, while some basic rules and a generic description are provided here, further rules may be specified for specific transport protocols within the appropriate Interoperability Profiles.

2.6.1. Root Layer Packet Format

The format of ACN packet shall consist of a preamble field, a single PDU block and a postamble field.

```
+-----+
| preamble |
+-----+
| PDU block |
+-----+
| postamble |
+-----+
```

Use of the preamble and postamble fields is specific to individual transport protocol.

2.6.1.1. Preamble

The format and content of the Preamble are dependent on the specific transport protocol and shall be defined in the appropriate interoperability profile. The Preamble may be used to hold any data which are necessary for the particular transport and may be empty.

e.g. on UDP systems, a marker string is placed in the Preamble to enable network analyzers and other diagnostic tools to easily identify ACN packets.

If the specification for transporting ACN on a particular protocol does not explicitly specify data to go in the Preamble field then this field shall be empty.

2.6.1.2. PDU Block

The PDU Block within a packet may contain multiple PDUs of the same protocol and/or PDUs of multiple protocols in any combination.

All the PDUs contained in the PDU Block shall follow the PDU format defined in this standard. How the Data field is formatted in these PDUs is up to the specification of the individual protocols. The protocols also define if the PDU contains a PDU block or other means of sub-dividing the contents of the PDU's Data field.

2.6.1.2.1. Vector Field in Root Layer PDUs

The Vector field of all Root Layer PDUs shall contain the Protocol Identifier of the protocol which generated the Data field within the PDU, formatted as defined in this specification.

2.6.1.2.2. Header Field in Root Layer PDUs

The Header field in Root Layer PDUs shall contain the CID of the component that generated the PDU (the Source CID).

2.6.1.2.3. Data Field in Root Layer PDUs

The Data Field in Root Layer PDUs shall contain protocol data which is opaque to the Root Layer Protocol.

2.6.1.3. Postamble

The format and content of the Postamble are dependent on the specific transport protocol and shall be defined in the appropriate interoperability profile. The Postamble may be used to hold any data which are necessary for the particular transport and may be empty.

e.g. on transport protocols which provide no error checking, the Postamble may be specified to hold a checksum for the packet.

If the specification for transporting ACN on a particular transport protocol does not explicitly specify data to go in the Postamble field then this field shall be empty.

The size of the preceding PDU block and therefore the location of the postamble is not specified here. Therefore any EPI which specifies the use of the postamble field shall also specify how it is located.

2.6.2. Root Layer Protocol Operation

On transmission the Root Layer Protocol shall collate PDUs from higher layer protocols into the PDU Block of a packet. It shall add Preamble and Postamble fields as defined by the specification for the particular transport protocol and shall pass the complete packet to that transport protocol for transmission.

On reception, the Root Layer Protocol shall first perform any validation or filtering operations using the Preamble or Postamble fields which are defined for the specific transport protocol used. If the packet is then accepted, it shall extract the PDUs from the packet and pass them on to appropriate protocol handlers in accordance with the PDU processing rules.

The Root Layer Protocol shall make the Source CID from the Header field of each PDU available to the higher layer protocols in whatever way is suitable or necessary for the particular implementation.

If a PDU is received whose ProtocolID (in the Vector field) is not recognized, the Root Layer Protocol may log this case but shall not stop processing the packet. It shall discard the unrecognized PDU and continue with the next (if any). The presence of the Length field in all PDUs allows the next PDU to be found with no further knowledge of the meaning of the current one.

Each Root Layer PDU shall only contain data from a single Component.

2.6.3. Root Layer Addressing

The Root Layer Protocol does not directly handle addressing of packets or PDUs and does not translate between Component Identifiers and transport layer addresses as might be expected - this is left to client protocols.

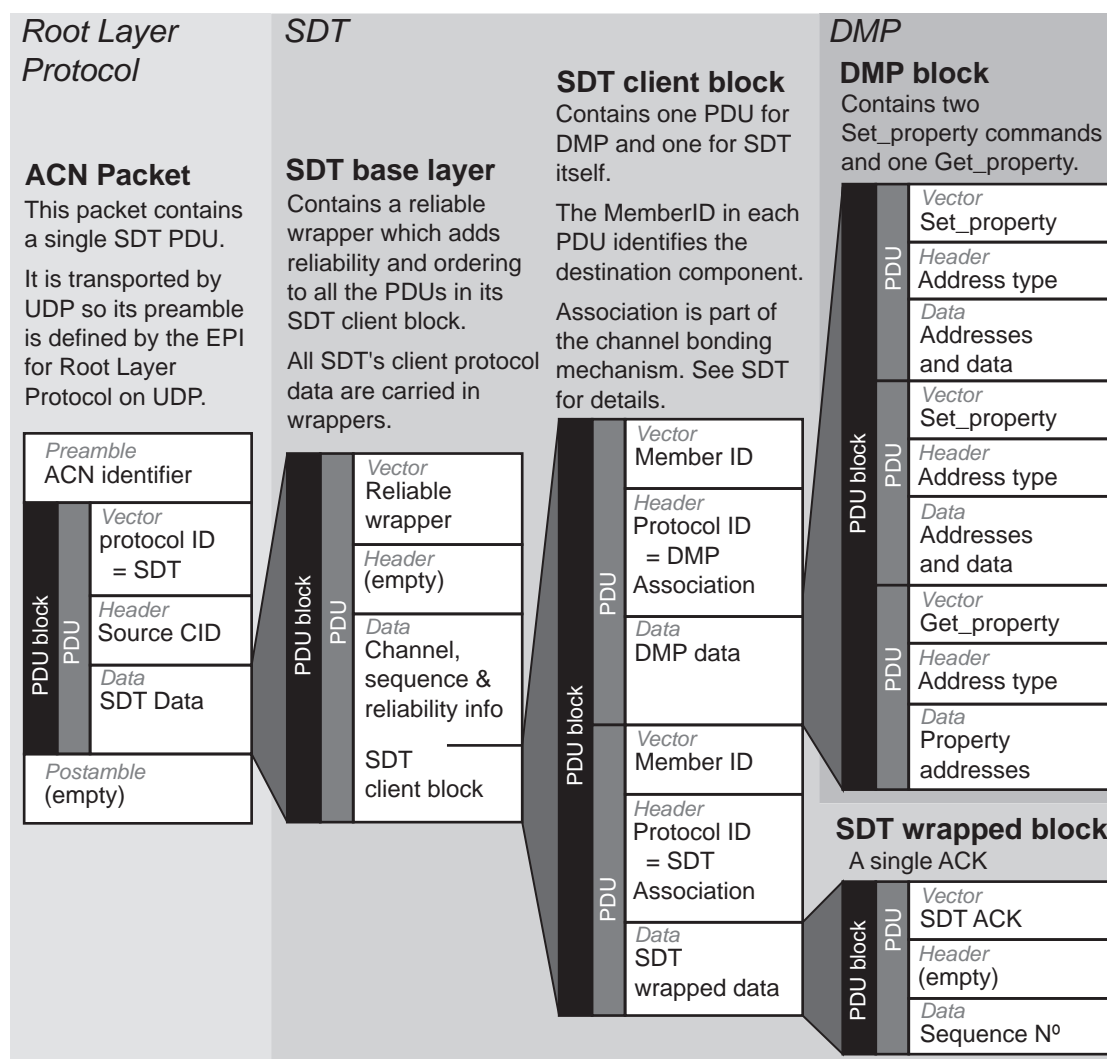
2.6.4. Byte Ordering in Root Layer Protocol

All fields within the ACN Root Layer Protocol PDUs shall use network byte order with the exception of the Data field which is passed to client protocols. The Data field is “opaque” to the Root Layer Protocol and its format - including byte ordering - is defined by the client protocol.

2.7. ACN PDU Structure Example

The example shows the structure of a packet which contains SDT and DMP messages.

Figure 4. Example of PDU Nesting in an ACN Packet



Note

This diagram is schematic only. It shows the values of the vector, Header and Data fields in each PDU, but does not show the Flags or Length and does not indicate whether values are direct or inherited.

At the two outer packing levels, each PDU block contains just one PDU. The SDT client block contains one PDU addressed to a DMP component and one containing an SDT message (in this case an ACK). Within the PDU passed to the DMP component is a PDU block interpreted by DMP which contains three DMP PDUs.

In accordance with Section 2.4.4, “Header” the size of header is fixed at each level except the DMP PDU block. In the Root Layer Protocol, Header always contains the source CID and is therefore 16 octets. At SDT's base layer and in SDT wrapped PDUs the header is empty (size = 0) while in an SDT client block header always contains a protocolID and association field (6 octets). In DMP the header size varies but is encoded in a consistent way within the first octet.

2.8. Protocol Identifiers

Protocols used within ACN shall be identified within packets using a 32-bit ProtocolID as defined in [ESTA-IDs].

ProtocolIDs shall be transferred in the Root Layer Protocol using all four octets in network byte order.

Unless an different format is explicitly specified within the definition of a higher level protocol, ProtocolIDs shall also be transferred using all four octets in network byte order wherever they occur within ACN messages.

2.9. Interoperability Profiles

The protocols defined in ACN follow a very modular design process and every attempt has been made to make modules independent of one another - this allows modules to be added, omitted or recombined as required. Furthermore within modules many parameters such as timeouts or maximum packet sizes are not fixed by any natural constraint but are necessarily dependent for optimal operation on the settings of other modules or on environmental factors such as network topology, latency or reliability requirements.

Because of this the core protocol suite does not define how interoperability is to be achieved in any particular application area. Barriers to interoperability exist at all levels from the specification of the plug (e.g. 10BASE-T vs 10BASE-2 Ethernet) right up to the highest level protocol used. ACN cannot and should not specify all these parameters as part of its core specification.

This problem is addressed in ACN by specification of Interoperability Profiles which specify what combinations of modules and parameters must be used to gain interoperable operation in a particular environment (e.g. for Device control over a Local Area Network using UDP/IP on Ethernet). An Interoperability profile shall specify values or acceptable ranges for all parameters defined within the ACN protocols it references and shall specify which additional protocols or facilities it relies upon for correct operation.

For example, the profile for Device control over IPv4 over Ethernet must specify which standards must be implemented and combined right from the Ethernet Layer, ancillary IP protocols (IGMP, ICMP, DHCP, ARP, IPv4, RIP, SLP, zeroconf IPv4LL or whatever) and parameters (MTU etc.) through SDT parameters (timeouts, retry counts) up to use of DMP.

An Interoperability Profile should also specify what additional facilities are required to link systems using it to systems operating other related profiles. This might range from a simple link layer bridge (e.g. Ethernet to IEEE 1394 - “Firewire”), to a higher layer protocol translation.

Additional Interoperability Profiles may be defined from scratch or by derivation from existing profiles as required.

2.9.1. Naming of Interoperability Profiles

Each Interoperability Profile shall be given a clearly distinguishable name which may be used in labeling of equipment which operates it. The specification of profiles should be such that users can be confident that two pieces of equipment which correctly implement it can be connected together and will interoperate - subject to any configuration requirements and constraints which must be clearly laid out within that profile.

Definitions

CID	Component Identifier. A UUID [UUID] identifying a particular component.
component	The process, program or application corresponding to a single ACN endpoint. All messages in ACN are sent and received by a component. See [Arch] for a more complete definition.
message	A valid PDU.
ordering	A mechanism that ensures that all messages within a session are processed by the members in the order intended by the leader.
packet	A PDU block sent as a distinct unit in the underlying protocol (e.g. a UDP datagram).
PDU	Protocol Data Unit is a single message (or command). A PDU may contain a PDU block with further embedded PDUs.
PDU block	A contiguous set of PDUs (messages) within a packet or containing PDU.

References

Normative

[ACN] Entertainment Services and Technology Association [<http://www.esta.org/tsp/>]. ANSI E1.17-2006. *Entertainment Technology - Architecture for Control Networks*. 2006-10-19.

[Arch] Entertainment Services and Technology Association [<http://www.esta.org/tsp/>]. ESTA TSP CP/2003-1007R4. *Entertainment Technology – Architecture for Control Networks*. “ACN” Architecture. 2006-10-19.

- [DMP] Entertainment Services and Technology Association [<http://www.esta.org/tsp/>]. ESTA TSP CP/2003-1010R3. *Entertainment Technology - Architecture for Control Networks*. Device Management Protocol. 2006-10-19.
- [SDT] Entertainment Services and Technology Association [<http://www.esta.org/tsp/>]. ESTA TSP CP/2003-1009R4. *Entertainment Technology - Architecture for Control Networks*. Session Data Transport Protocol. 2006-10-19.
- [DDL] Entertainment Services and Technology Association [<http://www.esta.org/tsp/>]. ESTA TSP CP/2003-1011R4. *Entertainment Technology - Architecture for Control Networks*. Device Description Language. 2006-10-19.
- [UDP] Internet Engineering Task Force (IETF) [<http://ietf.org/>]. RFC 768 [<http://ietf.org/rfc/rfc0768.txt>]. Postel. *User Datagram Protocol*. 1980.
- [ESTA-IDs] Entertainment Services and Technology Association [<http://www.esta.org/tsp/>]. ESTA TSP CP/2004-1027R3. *Entertainment Technology - Architecture for Control Networks*. EPI-16 ESTA Registered Names and Identifiers – Format and Procedure for Registration. 2006-10-19.
- [SLPv2] Internet Engineering Task Force (IETF) [<http://ietf.org/>]. RFC 2608 [<http://ietf.org/rfc/rfc2608.txt>]. Guttman, Perkins, Veizades, and Day. *Service Location Protocol, Version 2*. 1999.
- [RLP-UDP] Entertainment Services and Technology Association [<http://www.esta.org/tsp/>]. ESTA TSP CP/2004-1028R2. *Entertainment Technology - Architecture for Control Networks*. EPI 17. Root Layer Protocol Operation on UDP. 2006-10-19.
- [UUID] Internet Engineering Task Force (IETF) [<http://ietf.org/>]. RFC 4122 [<http://ietf.org/rfc/rfc4122.txt>]. P. Leach, M. Mealling, and R. Salz. *A Universally Unique Identifier (UUID) URN Namespace*. July 2005.
- [DMX512] Entertainment Services and Technology Association [<http://www.esta.org/tsp/>]. ANSI E1.11-2004. *USITT DMX512-A - Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories*. 2004.

ANSI E1.17-2010 Architecture for Control Networks Device Description Language (DDL)

*Part of ANSI E1.17-2010, a revision of ANSI E1.17-2006, approved by the
ANSI Board of Standards Review on 6 January 2011.*

Copyright © 2011 PLASA North America. All rights reserved.

TSP document CP/2009-1024r1

Revision History	
R5-draft-435	2009-07-22
R5pre3	06/17/09
R5pre2	2009-03
R5pre1	2008-03-14
R4	2006-03-14
R4pre2	2006-03-08
R4pre1	2006-03-03
R3	2005-10-14
R3pre2	2005-05-20
R3pre1	2005-04-25
R2	2004-11-09
R1	2004-10-07
Pre1	2003

Abstract

DDL is a language for describing devices controllable by getting and setting of properties (this can apply to almost anything!). this Standard describes a generalized control model for such devices. It specifies an XML based syntax for declaring the model for particular devices and defines how to use the language in conjunction with specific control protocols.

Table of Contents

1 Introduction.....	7
1.1 Origin and Goals.....	7
1.2 Modularity.....	7
1.3 Parts of DDL.....	7
2 Scope Purpose and Application.....	7
2.1 Scope.....	7
2.2 Purpose.....	8
2.3 Application.....	8
3 The Device Model.....	8
3.1 Device Model Introduction.....	8
3.2 Descriptions, Classes, Instances and Identifiers.....	8
3.3 Modules, Devices, and Appliances.....	8
3.4 Hierarchy of Properties.....	9
3.4.1 Modified Tree Structure.....	9
3.4.1.1 General.....	9
3.4.1.2 Branch Re-combination and Shared Properties.....	9
3.4.2 Property Values.....	9
3.4.2.1 General.....	9
3.4.2.2 Complex Values Harm Interoperability.....	9
3.4.3 Examples.....	10
3.4.4 Meaning of Property Values – Driven Properties.....	14
3.4.4.1 examples.....	14
3.4.4.2 Implied Properties.....	15
3.4.4.3 Unattainable targets.....	15
3.4.5 Order of properties – Chains of Control.....	15
3.5 Hierarchical Device Structure.....	15
3.6 The Access Protocol.....	16
3.6.1 Equipment Supporting Multiple Access Protocols.....	16
3.7 Four Locations for Property Value.....	16
3.7.1 Immediate Value Considerations.....	17
3.8 Grouping Properties.....	17
3.9 Property Behaviors.....	18
3.9.1 Multiple Behaviors of a Property.....	18
3.9.2 Behavior Definitions.....	18
3.9.2.1 Reading Behavior Specifications.....	19
3.9.2.2 Reading and Displaying Behaviors.....	19
3.9.2.3 Availability and use of Behavior Definitions.....	19
3.9.3 Behavior Derivation and Refinement.....	20
3.9.3.1 General.....	20
3.9.3.2 Multiple Derivations.....	20
3.9.4 Predefined Behavior.....	20
3.9.5 User Defined Behaviors.....	21
3.10 Labels.....	21
3.11 Property Values Types and Sizes.....	21
3.11.1 Value Types.....	21
3.11.2 Size is not Part of Value Type.....	21
4 DDL Syntax.....	21
4.1 Introduction.....	21
4.2 XML.....	22

ANSI E1.17-2010, ACN – Device Description Language (DDL)

4.2.1 Producers, Distribution and Consumers of DDL.....	22
4.2.2 Use of an XML Subset.....	22
4.2.2.1 Encoding and Detection of Encoding.....	22
4.2.2.2 Restrictions.....	22
4.2.2.3 XMLDecl.....	23
4.2.3 DTD, Schemas and Validation.....	23
4.2.3.1 DDL Document Type Definition (DTD).....	23
4.2.3.2 Use of Validation.....	23
4.2.3.3 Embedding Schemas in Documents.....	23
4.2.4 Namespace.....	23
4.3 Attribute and Element Value Types – XSD.....	24
4.4 DDL Modules – Devices, Behaviorsets and Languagesets.....	24
4.4.1 Module Content May not Change.....	24
4.4.2 Module Versions and Variations – alternatefor.....	24
4.4.3 Module Extension – extends.....	25
4.4.3.1 languageset.....	25
4.4.3.2 behaviorset.....	25
4.4.3.3 device.....	25
4.5 Root Element.....	25
4.6 Warning: Documents, Files and Resources.....	25
4.7 String Resources.....	25
4.7.1 Languages and Search Order.....	26
4.7.2 Identification of Strings.....	27
4.8 Arrays of Properties.....	27
4.8.1 Multidimensional Arrays.....	28
4.8.2 Properties That Do Not Iterate Across an Array.....	28
4.8.3 Immediate Values for Array Properties.....	28
4.8.4 Parametric Array Sizes.....	28
4.9 Element Identifiers.....	28
4.10 Use of XML Identifiers to Reference Device Model Features.....	29
4.10.1 Reference to Elements in Device Trees – Extended Reference Syntax.....	29
4.10.1.1 Examples.....	30
4.10.2 References in Arrays.....	30
4.10.2.1 Single Dimension Correspondence Match.....	30
4.10.2.2 Single Dimension Many-to-one Match.....	30
4.10.2.3 Multidimensional Arrays.....	31
4.10.2.3.1. multidimensional correspondence match.....	31
4.10.2.3.2. multidimensional many-to-one match.....	31
4.10.2.3.3. Notes.....	31
4.10.2.4 Array Matching String syntax.....	31
4.10.2.5 Default Matching for Multi-dimensional Arrays.....	32
4.10.2.6 Examples.....	32
4.11 Shared Properties.....	32
4.11.1 Defining-declaration.....	33
4.11.2 propertypointer.....	33
4.11.3 Special Case – Property Shared Across an Array.....	33
4.12 Declaration of of Sub Devices.....	34
4.12.1 Statically Included Device Descriptions.....	34
4.12.2 Sub Devices Attached by Reference.....	34
4.12.3 Circular References.....	35
4.12.4 Which Method to Use?.....	36
4.12.5 Discovery of Device Structure.....	36

ANSI E1.17-2010, ACN – Device Description Language (DDL)

4.13 Parametric Devices.....	36
4.13.1 Parameter Fields.....	36
4.13.2 Defining Parameter Names.....	37
4.13.3 Parameter Substitution.....	37
4.13.4 Restricting Parameter Values.....	37
4.13.5 Binding Fields to Parameters.....	37
4.13.6 Default Values.....	38
4.13.7 Attributes with Default Values.....	38
4.13.8 Template Example.....	38
4.13.9 Adoption of Parameters from Included Devices by Parent Devices.....	39
4.13.10 Re-parametrizing Supplied Values.....	39
4.13.11 Notes and Additional Rules.....	40
4.13.12 Dynamic use of Parametric Devices.....	40
4.14 Logical and Syntactic Property Structures.....	41
5 DDL Element and Attribute Reference.....	41
5.1 alternatefor.....	41
5.2 altlang.....	42
5.3 array.....	42
5.3.1 Arrays of Values.....	42
5.4 behavior.....	43
5.5 behaviordef.....	43
5.6 behaviorset.....	44
5.7 choice.....	44
5.8 date.....	44
5.9 DDL.....	45
5.10 device.....	45
5.11 extends.....	46
5.12 hd.....	46
5.13 includedev.....	46
5.14 key.....	47
5.15 label.....	47
5.16 lang.....	48
5.17 language.....	48
5.18 languageset.....	49
5.19 maxinclusive.....	49
5.20 mininclusive.....	50
5.21 name.....	50
5.22 p.....	51
5.23 parameter.....	51
5.24 .paramname.....	52
5.25 property.....	52
5.25.1 valuetype="NULL".....	53
5.25.2 valuetype="network".....	53
5.25.3 valuetype="implied".....	53
5.25.4 valuetype="immediate".....	53
5.26 propertypointer.....	54
5.27 protocol.....	54
5.28 provider.....	55
5.29 ref.....	55
5.30 refinement.....	55
5.31 refines.....	56
5.32 section.....	56

ANSI E1.17-2010, ACN – Device Description Language (DDL)

5.33 set.....	56
5.34 setparam.....	57
5.35 sharedefine.....	57
5.36 string.....	58
5.37 type.....	58
5.38 useprotocol.....	59
5.39 UUID.....	59
5.40 UUIDname.....	60
5.41 value.....	60
5.42 valuetype.....	61
5.43 version.....	61
5.44 xml:space.....	61
5.45 xml:id.....	62
6 Interface to the Access Protocol.....	62
6.1 Access to Network Properties.....	62
6.1.1 Network Property Values.....	62
6.1.2 Network Data Type and Size.....	62
6.2 Necessary Definitions and Restrictions.....	62
6.2.1 Protocol Declaration.....	62
6.2.2 Network Property Access Mappings.....	63
6.3 Behaviors.....	63
Appendix A. Generic DTD DDL.....	64
Appendix B. DDL Interface to DMP.....	69
B.1 Relationship Between DDL Devices and DMP Components.....	69
B.2 Message Mappings.....	69
B.3 DMP Property Access.....	69
B.3.1 Addressing.....	69
B.3.1.4 Relative Addressing and Location Origin.....	69
B.3.1.5 Absolute Addressing.....	70
B.3.1.6 Non-network Properties.....	70
B.3.2 Property Arrays.....	70
B.3.3 Accessibility.....	70
B.3.4 Property size and variable size properties.....	70
B.3.5 Additional Characteristics of DMP Properties.....	71
B.3.6 Byte Ordering.....	71
B.4 DMP Element and Attribute Reference.....	71
B.4.1 useprotocol (DMP devices).....	71
B.4.2 protocol (DMP devices).....	71
B.4.3 propref_DMP.....	72
B.4.4 childrule_DMP.....	72
B.4.5 loc.....	73
B.4.6 abs.....	73
B.4.7 inc.....	74
B.4.8 read.....	74
B.4.9 write.....	74
B.4.10 event.....	74
B.4.11 varsize.....	75
B.4.12 size.....	75
B.5 Examples.....	75
B.5.1 Simple Property.....	76
B.5.2 Array of Simple Properties.....	76
B.5.3 Included Device.....	76

ANSI E1.17-2010, ACN – Device Description Language (DDL)

B.5.4 Subdevice Attached by Reference.....	76
B.5.5 External Device References.....	77
B.5.6 Array of Sub Devices.....	77
B.5.7 Array of External Device References.....	78
B.5.8 Array Example.....	79
Appendix C. DTD for DMP Only.....	79
Appendix D. Languagesets and Behaviors Defined for DDL.....	85
Appendix E. Definitions.....	85
Appendix F. References.....	87

1 Introduction

Device Description Language is a language for describing controllable [devices](#) in a way that is machine readable and enables [controllers](#) to automate the process of interfacing to them.

1.1 Origin and Goals

Device Description Language (DDL) was developed as a an adjunct to the Device Management Protocol [DMP] that is a part of the E1.17 Network suite developed by the Entertainment Services and Technology Association (ESTA) [ACN]. (ESTA became PLASA North America effective 1 January 2011.)

The design goals for DDL are:

1. DDL should describe devices rather than dictate their functionality and modality.
2. The designer of controlled equipment must be able to describe their device and all its features in as much depth of detail as possible.
3. Any process interpreting a description need not “understand” all of it but must be able to access and handle as much or as little as its designer chooses.
4. DDL must keep the human centric view of what equipment does separate from the algorithmic view of how to control it.
5. DDL should provide a generic language capable of describing devices not yet invented.
6. A clear path for extensibility should be provided.
7. Device descriptions should be static so they can be passed around with or without the equipment they describe and so that identical devices have identical descriptions.
8. DDL should leverage whatever technologies are already available.
9. The burden of policing conformance and providing extensions to the specification should be minimized.

A consequence of 2 and 3 is that interoperability between controller and controlled equipment should reach the highest level of control achievable by the less capable of the two. It is important to bear in mind when reading this specification that controllers are not required to support or act on all the information contained within a device description. Simple controllers will look for a few basic elements while complex controllers will go to greater depth.

1.2 Modularity

To encourage re-use and commonality of descriptions and to extend its flexibility, DDL is a modular language. This means that a single piece of equipment may be described by multiple linked description modules and allows modules describing common functional blocks to be re-used directly in other equipment.

1.3 Parts of DDL

There are three major parts to DDL:

1. The device model – an abstract model used for describing devices.
2. The XML format for DDL – a syntax for rendering device descriptions in a machine and human readable serial format.
3. The Protocol interface – how device descriptions marry up to a particular protocol.

2 Scope Purpose and Application

2.1 Scope

This standard defines a generalized description language for devices that may be controlled remotely via a

data link or networking protocol. DDL does not, for example, describe controller functions although control equipment frequently contains parts that are remotely controllable or monitorable and so some aspects of that equipment might have DDL descriptions.

2.2 Purpose

DDL has been developed together with a protocol suite aimed at control of entertainment technology equipment – motion control, lighting and sound level control, timed sequences etc. – The ESTA ACN Protocol Suite [ACN]. The DDL standard was developed to facilitate a high degree of automation in the configuration of control networks, and in particular in the adaptation of controllers to the controllable devices that are discovered in an arbitrary network.

2.3 Application

While DDL was developed as part of the ACN protocol suite, it has potential application in many control networks. It is directly suited to devices that may be controlled by reading and writing to registers or locations within the device using few message types. Protocols in which devices are controlled by complex messages to few destinations may require some transformation to fit with the DDL model.

3 The Device Model

3.1 Device Model Introduction

The device model provides the underlying structure to DDL. It provides a way to think about how devices are broken down step by step into their fundamental control elements. Having analyzed the device in this way, the hierarchy is laid out in a file or files in a text format (the syntax).

DDL is a hierarchical modular language and in DDL the term device means the piece of equipment described by a single module including all the modules contained within it. So if multiple modules (each describing a device) are combined into a larger module, then that too describes a device.

3.2 Descriptions, Classes, Instances and Identifiers

Each device description describes a [class](#) of devices. Any physical device that conforms to that description is then called an instance of that class. DDL uses a UUID [UUID] to identify device classes and by extension their DDL descriptions. Any UUID that identifies a device class is called a Device Class Identifier or [DCID](#).

3.3 Modules, Devices, and Appliances

DDL [devices](#) are modules that may describe an entire piece of equipment or may just describe a part of it. DDL allows any device to contain sub-devices (also called child devices) and there is no distinction between devices and sub-devices in the way that DDL describes them – any device may be included as a child device of some parent.

Any instance of a device that has no parent device is called a [root device](#). The entire device structure formed by a root device and all its children and descendants is called an [appliance](#). An appliance typically corresponds to a single piece of equipment and is frequently a single network entity although this is dependent on the protocol used. (In ACN systems using DMP an appliance corresponds to a [component](#) that exposes properties. Confusingly, DMP calls this a device (see: Relationship Between DDL Devices and DMP Components)

While a DDL device is a module, DDL syntax also defines other modules that are not devices (see: DDL Modules – Devices, Behaviorsets and Languagesets).

3.4 Hierarchy of Properties

3.4.1 Modified Tree Structure

3.4.1.1 General

The hierarchy of properties representing a device in DDL forms a set of modified tree structures. In the DDL device model each node of the tree is called a property and one or more trees of properties represents the structure of a device (the model is a set rather than a single tree, because there can be multiple root-level properties in a device).

A tree is a structure very familiar in computing and consists of a set of nodes connected such that every node is the child of a parent node. The exception is a single node in each tree called the root node that has no parent itself but is the parent, or more distant ancestor of all other nodes within the tree. Any node may have any number (including zero) of child nodes and a node with no children is called a leaf node.

3.4.1.2 Branch Re-combination and Shared Properties

The structure of DDL is modified because separate branches of the tree may re-combine – in which case the node at a re-combination point has multiple parents. The re-combination is restricted in DDL: it is a requirement that there shall be no property that is both ancestor and descendent of any other property or of itself. In mathematical terms this is a directed acyclic graph or DAG [http://en.wikipedia.org/wiki/Directed_acyclic_graph].

For example, consider an automated lighting fixture with an initialization operation that is performed by setting the Boolean value of an initializer property. Structurally the initializer is a sub property of the property representing the function that it initializes (e.g. an initializer for the pan function would be a sub property of the pan property). But it is common practice in real devices for a single initializer to cause initialization of multiple functions of the device (one initializer for both pan and tilt). This means that the one initializer is shared between pan and tilt. In DDL this situation is allowed – the initializer property has both pan and tilt as its parents so the pan and tilt branches of the tree re-combine at the initializer property.

In the XML syntax defined below, there is no direct way to represent re-combination of the branches of the tree, so this is achieved by declaring the same property within each of its parents and indicating that it is shared. The shared property is fully declared in one place only – this is the defining declaration – and in all other places that the same property needs to appear, a simple reference to the defining declaration is used – this is called a propertypointer (see: Shared Properties).

3.4.2 Property Values

3.4.2.1 General

As well as zero or more child properties (also called sub properties or properties of properties) each property shall have zero or one values.

A property value is a single item that may be a number, a text string or sometimes a more specialized object such as an Ethernet MAC address. Typically the value may be read or written via the network, although it may also be implied or provided within the DDL itself.

3.4.2.2 Complex Values Harm Interoperability

Any structure within a property value is not visible to the device model. Creation of devices with properties whose values have complex structure is strongly discouraged – the philosophy of creating devices that can be described using a common language like DDL conflicts with attempts to bury complex structure and behavior within the values themselves and the chances of a generic controller being able to handle such

values is greatly reduced.

3.4.3 Examples

For example consider a device that is a remote controlled robotic camera. This might have a property for exposure control, one for an X,Y,Z positioner and one for the pan-tilt-rotate direction:

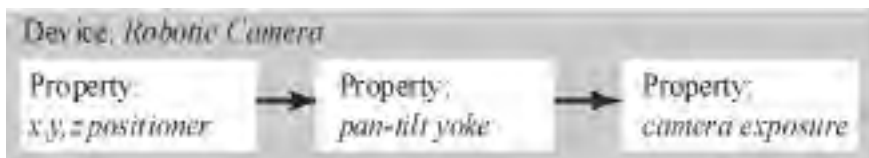
XML and DMP examples

Snippets of XML description are included in examples throughout this section of this Standard. These examples are often simplified for readability and may not conform strictly to the DDL specification.

An ellipsis, “...” is used in many examples to indicate which portions not relevant to the example have been omitted for brevity. This is not part of the XML syntax.

Examples also use ESTA DMP [DMP] extensively. The reader should be aware that DDL can be used with other protocols and in fact that the device model is not tightly coupled to XML and could be represented using other syntax (although the current project has no plans to do so).

Figure 1: Robotic Camera Property Layout



Example 1: Robotic Camera Sample DDL

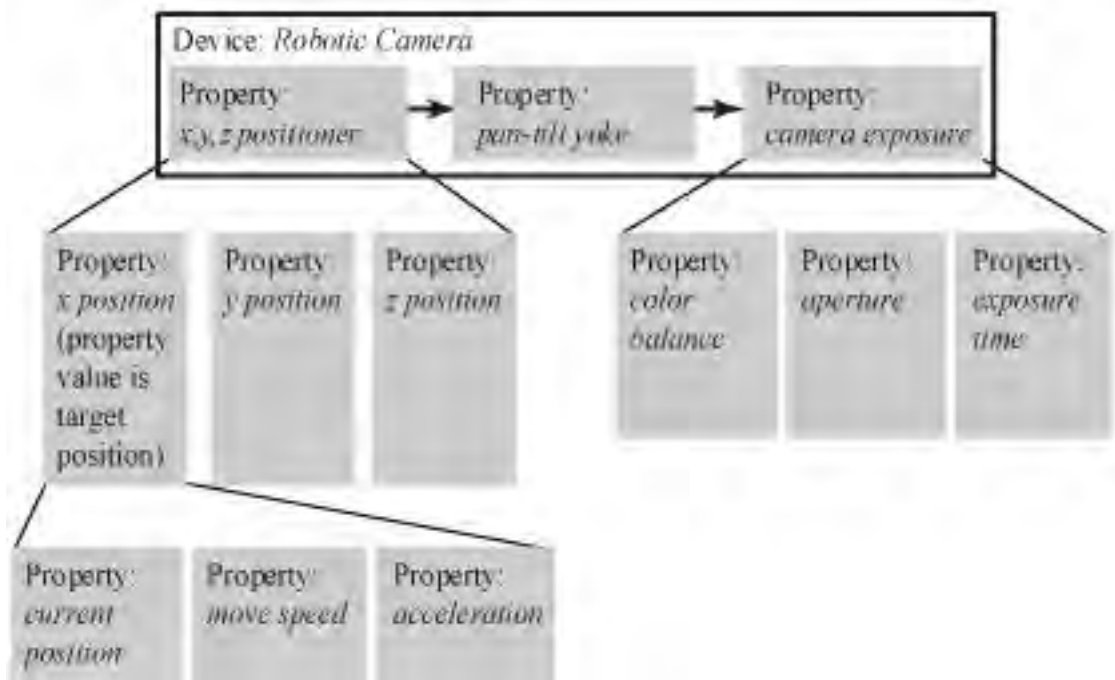
```

<!-- description of a robotic camera -->
<property ...>
  <label>Robotic Camera</label>...
  <property ...>
    <label>x,y,z positioner</label>...
  </property>
  <property ...>
    <label>pan-tilt yoke</label>...
  </property>

  <property ...>
    <label>exposure control</label>...
  </property>
</property>
  
```

In their turn X, Y and Z are separate child properties of the positioner that have values representing the position for X, Y and Z. Each of these motor driven axes in turn might have a child property representing desired or target position, maximum speed, acceleration or other parameters used to control movement. The speed property would probably have child properties that define its upper and lower limits. Other properties of X, Y and Z might be the units they operate in (meters, millimeters etc.). The exposure control meanwhile could have a property for color balance and one for exposure time/aperture.

Figure 2: Robotic Camera Extended Property Layout



Example 2: Robotic Camera Extended Sample DDL

```

<DDL>
<device>
  <!-- description of a robotic camera -->
  <property ...>
    <label>Robotic Camera</label>...
  <property ...>
    <label>x,y,z positioner</label>...
  <property ...>
    <label>X position</label>...
    <protocol name="example">...</protocol>
  <property ...>
    <label>target position</label>...
    <protocol name="example">...</protocol>
  </property>
  <property ...>
    <label>speed limit</label>...
    <protocol name="example">...</protocol>
  </property>
  <property ...>
    <label>acceleration</label>...
    <protocol name="example">...</protocol>
  </property>
</property>
<property ...>
  <label>Y position</label>...
  <!-- expands similarly to X axis -->
</property>
<property ...>

```

```

    <label>Z position</label>...
    <!-- expands similarly to X axis -->
  </property>
</property>
<property ...>
  <label>pan-tilt yoke</label>...
</property>
<property ...>
  <label>exposure control</label>...
  <property ...>
    <label>color balance</label>...
  </property>
  <property ...>
    <label>aperture</label>...
  </property>
  <property ...>
    <label>exposure time</label>...
  </property>
</property>
</device>
</DDL>

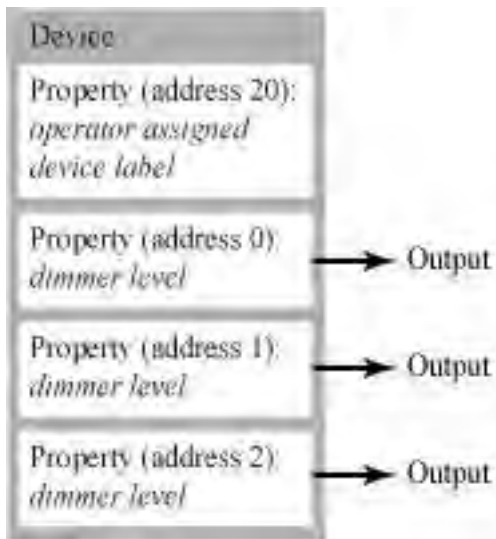
```

At the higher levels in the tree hierarchy properties represent major functional blocks of the device (such as the X,Y,Z positioner in the example above) and at the very top of each tree there is always just one root property (logical trees are usually viewed upside down with the root at the top), in this case representing the camera itself (a DDL device may however contain more than one tree). At the highest levels properties often do not have a value but are simply containers for other properties. As the hierarchy descends properties may represent smaller items (e.g. the desired X position) and at the lowest level they are points of detail such as the upper limit on the speed setting for a motion control (which would probably be a fixed value present for informational purposes).

The control structure of the device is represented by the structure and attributes of properties and the behavior they provide. The state of the device is represented by the values associated with the properties. To control or operate the device, the values of its properties must be changed.

Figure 3: shows an example of a very simple three-way dimmer pack. There is just one property for each dimmer that represents the dimmer output level. There is also an additional property that is a non-volatile string that the operator may use to store a label for this particular pack. The label attribute is a simple text label that could be presented to the user to tell them what purpose the property serves.

Figure 3: Simple Dimmer Pack Property Layout



Example 3: Simple Dimmer Pack Sample DDL

```

<DDL>
  <device>
    <property ...><label>Dimmer pack</label>...
    <property ...>
      <label>Device supervisory stuff</label>...
      ...
    <property ...>
      <label>Device Label</label>...
      <protocol name="example">
        <address persistent="true">20</address>
      </protocol>
    </property>
  </property>
  <property ...>
    <label>Dimmer 1</label>...
    <property ...>
      <label>Dimmer 1 Intensity</label>...
      <protocol name="example"><address>0</address></protocol>
    </property>
  </property>
  <property ...>
    <label>Dimmer 2</label>...
    <property ...>
      <label>Dimmer 2 Intensity</label>...
      <protocol name="example"><address>1</address></protocol>
    </property>
  </property>
  <property ...>
    <label>Dimmer 3</label>...
    <property ...>
      <label>Dimmer 3 Intensity</label>...
      <protocol name="example"><address>2</address></protocol>
    </property>
  </property>

```

```

    </property>
  </property>
</device>
</DDL>

```

3.4.4 Meaning of Property Values – Driven Properties

For many device features, the simplest control model, would appear to be to represent the state of that feature by a writable property such that when that property is changed, the software and hardware within the device adjust the state of the feature to match. However, in real devices there are almost invariably times when the state of the feature does not match the value written to the property. This can occur during transient states while the device is changing the feature – for example with a motor driven feature it can take significant time to reach a given target position; it occurs when there are values of the property that represent states that the feature cannot take – for example due to range limits; and it occurs when there are other inputs that may change that feature such as local controls or overrides or physical conditions such as obstructions (the scenery hitting the device).

As control models become more sophisticated, it is often the case that the feature is not controlled by a single property value, but by a combination of values – for example, a motor driven feature (such as pan or tilt for a projector or camera) is typically controlled internally using a combination of acceleration up to some speed limit, a period of movement at that speed and a deceleration to rest at the target. The device may expose all four parameters (acceleration, speed, deceleration, target) that all contribute to the dynamic control of the single movement axis.

In all these cases, where the interface is bidirectional, the designer may well wish to expose a read-only property that represents the actual state of a feature, in addition to the writable properties used to control it – this presents desirable feedback.

In order to present a consistent structure for modeling devices across this wide range of conditions DDL uses a rule that for any feature, there must be an outermost property that represents the actual state of that feature. This property then contains all the writable properties that are available to control that feature. In a bidirectional network where the device exposes the actual state of the feature, this is the outermost property in the description. However, in cases where there is no actual state value on the network, the description still includes an actual state property but must make it an implied property (see below) – indicating that the device model is identical but the actual state is not available.

This principle is extended within descriptions to the general case that one property may be driven by one or more driver properties. In which case, the driven property is always the outermost (closest to the root of the description tree) and the driver properties are child (or descendent) properties of it. As well as the case of driven actual values that differ dynamically from their target driver properties, driven properties are used extensively in DDL to represent cases where multiple properties are used in combination to achieve a result. Examples include cases where multiple network inputs contribute to some algorithmic process to achieve a result (e.g. “Highest Takes Precedence” combination common in lighting controls) and also selectors where a predefined restricted set of values are available and a property selects between them (e.g., an input voltage selector that can be set to select 100V, 115V or 230V).

3.4.4.1 examples

A property represents the desired or target position of a winch. When it is set to a new value, the winch controller must ramp up the motor speed, move for some time at a given speed and then ramp down the speed to come to rest at the desired position. While this is happening, the state of the winch is not that set in the property but is changing and a property representing the actual winch position would constantly change to represent this. This latter property representing the actual position could not be writable.

A lighting dimmer has 8 inputs each of equal weight, which are combined on a “highest takes precedence” basis (the dimmer adopts the maximum value out of the inputs). In this case, there is not just one target property but several drivers and their parent driven property reflects the highest value of all the drivers.

A manual slider is represented as a single property value that reflects the state of the slider. Because this feature is an input only and there is no way to control it, the actual value property contains no drivers at all.

3.4.4.2 Implied Properties

In some cases the value of a driven property may not be available – this can occur simply because the data link is unidirectional or because the device designer cannot or chooses not to expose it. For example, in an analog servo driven system the controller itself may have no feedback of actual position. In these cases the property is neither writable nor readable – this is called an implied property. Implied properties are present in the description so that the description structure is consistent, but is not present on the network. In some cases the value of an implied property may be inferred from the properties driving it. Implied properties must however be declared in the DDL where relevant as this is how DDL builds a consistent description of the device. In devices with a bidirectional interface, implementers should not use implied properties if a network readable value could be provided.

3.4.4.3 Unattainable targets

If a target state is set (by writing a value to a property) that is a legal operation within the protocol, yet that breaks constraints on that property (e.g. a value that exceeds a limit, or that would require movement at an unachievable speed), the device has two possible courses of action. It may reject the command and maintain its previous state, or it may do the best it can by adopting an attainable target that is close to that requested. DDL imposes no requirement on which choice is taken. However, access protocols (The Access Protocol) may define their own rules or may provide messages to handle this situation. Property behaviors may also be defined that allow control or expression of which option is taken or that show the attainable target that has been adopted.

3.4.5 Order of properties – Chains of Control

There are many devices that can be described as a set of operations that are performed in turn on a stream or channel of material. For example, audio channels are routinely subjected to a distinct sequence of operations (input gain, frequency filtering, effects, output gain etc.). Another example is an automated light where a sequence of operations are performed on a light beam (intensity, shutters, gobo, color filter, focus, effects etc.).

In cases where order of processing in such a channel would have an effect on the final outcome or on the value of intermediate properties (e.g. audio metering points), these properties shall be declared in the order in which they are applied to the channel. For more detailed analysis of these situations see stream behaviors in the ACN base behaviorset [acnbaseDDL].

3.5 Hierarchical Device Structure

There are many cases where describing an entire piece of equipment as a single device is not sensible or convenient. This occurs both because common items may repeat in many places (reuse of descriptions is beneficial both in saved space and in allowing controllers to recognise common structures) and because much equipment is modular and parts may be changed, either statically as when different options are present, or dynamically as when different modules are fitted or different configurations selected.

As seen in Modules, Devices, and Appliances any DDL device may contain child devices and may itself be contained within a parent. The partitioning of appliances into devices is at the choice of the author of the descriptions and DDL devices do not necessarily bear a close relationship to physically identifiable modules

of the equipment (although they frequently do so).

Sub devices can occur anywhere within the property tree of a device. Each root property and associated property tree of the sub-device is then considered to be a branch of the parent device tree in exactly the same order as they occur in the sub device. Mechanisms that allow the sub device type to change or be re-specified are very powerful in describing equipment that may change as modules are added or removed or as a result of reconfiguration.

The syntax mechanisms for declaring sub devices are specified in the syntax section.

3.6 The Access Protocol

The device model represents the state of a device by the values of its properties. The network provides the means to access and modify the values of those properties and the [access protocol](#) is the link between the control or device application and the network. (In the current implementation of DDL ESTA-DMP [DMP] is the only access protocol defined).

The value of a property is accessed by reading it (or writing where appropriate) via the access protocol. The device description needs to give enough information for a [controller](#) to know where and how the value is accessed. This information, which is highly dependent on the access protocol used, is contained in a [protocol](#) reference that is part of the property declaration within the description. The [protocol](#) reference shall identify the access protocol and then all the protocol-dependent information needed to access the value using that protocol (e.g. the address or message type, the data size and so on).

For any access protocol to be used with DDL an interface must be defined that specifies how the device model is interfaced to the access protocol, how this is represented in the syntax and any extensions needed to make DDL work with it. The requirements of this interface are described in Interface to the Access Protocol.

3.6.1 Equipment Supporting Multiple Access Protocols

When a piece of equipment supports multiple access protocols, the representation of that equipment from a protocol view-point may or may not allow a single common device model to be used for more than one of the protocols. There are also times when two or more protocols must be used in combination to effect control over the equipment. (e.g. a combination of [RDM] and [DMX512]).

Where a common device model is used for multiple protocols, then each property of the model may contain a separate [protocol](#) reference for each protocol by which it is accessible.

Where the protocols are independent and a common device model is not suitable, then separate and independent descriptions must be used.

3.7 Four Locations for Property Value

There are four “kinds” of property that are distinguished by where and how their value, if any, is located.

NULL

We have seen (see: Hierarchy of Properties) that a property need not have a value at all – it may simply have child properties. This is a NULL value property.

Network

This is the common case where the value of a property is accessible (readable and/or writable) via the network (see: The Access Protocol).

Implied

An implied property is one whose presence is implied by the functional features of the device but whose value is not directly accessible and must be inferred. (see: Implied

Properties)

Immediate

The fourth kind of property value is an immediate value. This is a value that is constant both over time and across all instances of the device. In this case the value may be provided directly in the DDL and is immediately available when reading the description.

3.7.1 Immediate Value Considerations

Before putting an immediate value into a description, a device designer must ask not only whether the value is the same for all instances of the device, but also whether the description may, in the future, be applied to new devices that may have a different value. For example, putting a product's model name into a description as an immediate value prevents the description being re-used if the model name changes (e.g. for marketing reasons). The availability of template devices and parameters can be used to mitigate this issue, but by using a Protocol reference for the model name marketing or geographical variations will not affect the description – on the other hand, the model name would not then be available to an offline program that had only the description available and no actual hardware.

Either course of action may have the advantage depending on the circumstances.

3.8 Grouping Properties

The relationships of properties representing states of a device cannot be expressed purely in the behavioral descriptions of those properties but are also expressed by the structural relationships of those properties in the description. Properties representing actual states of a device shall be grouped together in a way that directly models the logical way in which the control entities they describe relate within the device.

In the previous dimmer example each property is independent of the others and so each appears in its own group. If each channel of the device included a dimmer and a color changer, then these would appear together in the same group:

Example 4: Grouping Sample DDL

```
<DDL>
<device>
  <property ...>
    <label>colored lights</label>...
  <property ...>
    <label>Luminaire 1</label>...
    <property ...>
      <label>Intensity</label>...
      <protocol name="example"><location>101</location></protocol>
    </property>
    <property ...>
      <label>Color</label>...
      <protocol name="example"><location>102</location></protocol>
    </property>
  </property>
  <property ...>
    <label>Luminaire 2</label>...
    <property ...>
      <label>Intensity</label>...
      <protocol name="example"><location>201</location></protocol>
    </property>
```

```

    <property ...>
      <label>Color</label>...
      <protocol name="example"><location>202</location></protocol>
    </property>
  </property>
  <property ...>
    <label>Luminaire 3</label>...
    <property ...>
      <label>Intensity</label>...
      <protocol name="example"><location>301</location></protocol>
    </property>
    <property ...>
      <label>Color</label>...
      <protocol name="example"><location>302</location></protocol>
    </property>
  </property>
</device>
</DDL>

```

3.9 Property Behaviors

Simply listing the properties of a device – even in a structured way – does not provide the algorithmic information on behavior that a controller might need.

For example, consider a property representing the “theta” axis in a polar coordinate positioning system. Two (sub) properties of the theta axis are its scale and units (e.g. “0.1” and “degree”) others are its minimum and maximum values (e.g. “-200” and “+200”). The meaning of each property is expressed as one or more behaviors. All properties in DDL must have one or more associated behaviors to indicate their use. Each behavior expresses some aspect of what that property means and how it should be used.

In practical terms, each behavior attached to a property is simply a reference (by name) to a separate behavior definition that provides a full description and specification for any property with the named behavior.

3.9.1 Multiple Behaviors of a Property

A property may have multiple behaviors. This is required because the semantics of many behaviors are independent of each other. For example, a property may declare both “minimum limit” and a “floating point” behavior.

In some cases, multiple behaviors may be applied because they relate to slightly differing device models or to different access protocols. Implementers of DDL consuming applications (controllers in the widest sense) should be aware of this since it means that a property may carry behaviors that are not relevant to the scope of a particular application.

3.9.2 Behavior Definitions

A behavior definition describes in precise terms what any property that references it represents, provides a specification of how it must behave and lists rules or constraints on its value and usage. It may also specify contextual constraints such as which children it must contain or which behavior(s) its parent property must have (for example the maximum limit behavior is a constraint on its parent property that must have some sort of numerical value).

A DDL behavior definition shall associate three pieces of information.

- A name by which a controller can recognize that behavior and that is used to reference this behavior wherever it is applied to a property.
- A derivation that declares specific, more fundamental, or abstract behaviors from which this is refined by a process of specialization.
- A specification that defines the [semantics](#) of the behavior – what a property with this behavior means or does – and specifies any applicable constraints.

3.9.2.1 Reading Behavior Specifications

The specification part of a Behavior definition is written in structured text and is intended primarily to be read and understood by designers creating controller applications to enable them to write the necessary code to handle any special functionality they wish to implement to support the behavior.

A secondary target of the text is a user. While users cannot normally reprogram controllers to support specialized behavior, the description may assist them in controlling the equipment using whatever interface or facilities are available to them or guide them in assigning properties to particular control surfaces.

In view of the dual readership of the description part of a behavior declaration, it should be written carefully with an initial overview section followed by unambiguous technical detail.

3.9.2.2 Reading and Displaying Behaviors

The syntax of the DDL behavior element provides for structuring of the description text using paragraph elements arranged in nestable sections with headings. However, reading raw XML is not for the faint hearted and is certainly not expected to be imposed on the ordinary user.

There are a wealth of tools available for managing, manipulating and displaying XML and Behavior definitions – as with all of DDL – are best viewed using suitable tools rather than attempting to read the XML directly. A simple method is to develop a stylesheet that a browser can use to present an XML document in a readable form. However, off-the-shelf XML tools can be configured to go much further, for instance by extracting individual behavior descriptions, formatting them according to language and other preferences and displaying them within browsers or other applications.

A version of the behavior module transformed into HTML using XSL [XSL] is provided for easier reading. This version is not normative.

3.9.2.3 Availability and use of Behavior Definitions

In order to correctly interpret DDL, the behavior definitions must be available for not only the behaviors used within the device, but recursively for all the behaviors from which those are refined down to the most basic level. The mechanism for making DDL modules available depends on the rules governing generation and publication for particular protocols or environments. These rules should ensure that all necessary parts are made available together.

It is the choice of the controller designer to decide the extent to which individual behaviors are recognized and acted on when parsing a description.

For example in the polar axis scenario above:

- A very simple controller for diagnostic purposes could recognize only the base behaviors scalar and string. It could also present the additional behavioral information to the user if asked. The user, sees simply a set of scalars and strings with descriptive information available, and is able to control them.

- A more sophisticated controller could recognize the minimum and maximum as constraints and use them for example to scale a slider between the two and to label the display.
- A mid range lighting controller recognizes theta as well and knows to assign it to a trackball's X-axis.
- Finally, the most sophisticated controller confronted with the same description, not only recognizes theta but understands its meaning well enough to combine it with scale and units properties to do polar to rectangular transformations and present to the user in x,y,z form.

3.9.3 Behavior Derivation and Refinement

3.9.3.1 General

Any Behavior definition should specify one or more other behaviors that it refines and is derived from. This is analogous to object inheritance in programming languages that allows the creation of classes of behavior based on generic or abstract concept behaviors that are progressively refined to more specialist ones. Thus, for example, the maximum behavior mentioned above would be derived from a more generic limit behavior whilst other more specific behaviors could then be refined from maximum if required (e.g. to distinguish between inclusive and exclusive limits).

A controller encountering but not recognising a very specific behavior may yet have some provision for the more generic concepts from which the generic behavior derives. For example, a property with behavior representing rotation about a specific axis, refines the more generic behavior of rotation about an arbitrary axis. By tracing the refinement backwards from the specific, a less dedicated but nonetheless useful control approach may be available.

It is not required that a behavior definition be refined from pre-existing behaviors, but there are already a rich set of behaviors available [acnbaseDDL] and it is strongly recommended that wherever possible, new behavior definitions should be refined from these, as this improves the coherence and consistency of approach to control across devices and greatly improves the chances that a device containing new behaviors will be controllable by controllers from other manufacturers. This applies both to existing controllers, which may be able to provide partial control based on the behaviors from which the new ones are derived, and to future controllers whose designers are more likely to add provision for behaviors that fit within existing paradigms.

3.9.3.2 Multiple Derivations

A behavior may be a refinement of more than one other behavior, particularly where those others represent independent concepts. The derived behavior is then a combination of the two or more behaviors it refines (with additional meaning as declared in its specification).

For example, a behavior that represents a label string refines not only the more generic behavior of labeling (associating a human readable annotation with another property), but also the datatype behavior associated with strings. Alternative refinements of the labeling concept might combine it with a referencing behavior that retrieves the label text via an index, or with a pictorial behavior.

3.9.4 Predefined Behavior

The DDL specification itself (this document) does not define any behaviors. Each protocol using DDL may define behaviors in its specification that are appropriate to its access methods and the kind of equipment it may be required to control.

All protocol specifications must define at least one device reference behavior if they are to allow modular construction of devices (see: Declaration of of Sub Devices) other than by static inclusion.

There is a rich set of behaviors defined as part of the ESTA ACN Standardization. While some are DMP specific, most of these behaviors may be used for almost any protocol ([acnbaseDDL]).

3.9.5 User Defined Behaviors

Additional behaviors may be declared as required by DDL authors. The corresponding definitions shall be made available according to the rules of the protocol or environment in which they are used.

It is a goal that the set of defined behaviors that the designer of a controller may need to support is kept as small as possible. Authors of DDL should make every effort to use existing behaviors where possible. If defining new behaviors they should seek consensus among users with similar requirements and should derive them as refinements of existing behaviors. When creating new behaviors that are very different from existing ones, authors should also create intermediate behaviors in steps of refinement that seek to preserve as much generality and commonality with other applications as possible at each stage of refinement.

Devices using little-known behaviors that do not extend existing better known behaviors are likely to have greatly reduced interoperability.

3.10 Labels

A label may be attached to many elements and is intended as an indication of the function of that element to the end user or operator. It is entirely up to a controller to choose whether, when and how to present labels to the operator. Since labels are optional and their text is completely free, they are not suitable for use by a controller or other automated system in making algorithmic decisions.

Labels may either contain directly supplied text or may reference string resources (see: String Resources), which provides a mechanism for multilingual labeling.

3.11 Property Values Types and Sizes

3.11.1 Value Types

The interpretation of a property value depends on the behavior associated with it, while the encoding and size are largely a matter of the protocol used to access it (see: Property Values). When an immediate constant value is supplied, it is a raw text string (in the XML serialization) and the processing application must know how to parse it. This information must be implicitly or explicitly provided.

3.11.2 Size is not Part of Value Type

Data size is not necessarily given with a value type. This is because it is up to the controller to choose its own data representations, because sizing may vary from protocol to protocol and because size information may be misleading. Size may be specified as part of the protocol reference that may also specify a low level protocol data type if necessary to distinguish the encoding “on the wire”.

4 DDL Syntax

4.1 Introduction

The DDL syntax uses XML [XML] as a structure for rendering the device model in a textual format. This format goes beyond simply rendering the device model as it also provides the means to structure the documentation of devices in a modular fashion.

Because the device model is abstract, it would be possible to represent it using different formats and it is possible that alternatives may be developed in the future.

Note

Unrecognized elements in DDL shall be ignored by DDL interpreters. Future revisions of the DDL specification may add additional elements that must be ignored by implementations of earlier versions.

4.2 XML

Extensible Mark-up Language provides a framework for defining specific mark-up languages – DDL is such a language. The reader should be familiar not only with the XML standard [XML] but also with associated standards that are called out by this specification that makes no attempt to define or introduce XML beyond referring to these standards.

Note

Rules presented here that define restrictions on how XML may be used, apply to DDL documents as presented in the context of this standard for interoperable exchange between device and controller – this defines the flavor of XML that a controller or other consumer of DDL should expect to encounter whether those descriptions come from the device themselves, from manufacturer's sources or from elsewhere. Syntax restrictions (if any) for internal use within an application, for private use within an organization or for purposes standardized elsewhere are beyond the scope of this Standard.

4.2.1 Producers, Distribution and Consumers of DDL

DDL is *produced* by manufacturers of equipment and any other person or organization wishing to describe devices using the language.

It is *distributed* by many means including, but not limited to: supply over a network by the device itself, supply over the internet by the manufacturer or other organization, supply on transportable media (CD-ROM, USB-stick, Floppy disk etc.).

DDL is *consumed* by any application that reads and interprets one or more DDL descriptions. This includes controllers and monitors of equipment, applications presenting descriptions or parts thereof for human readers and applications transforming DDL into other forms.

4.2.2 Use of an XML Subset

While XML has the benefit of being a widely used off-the-shelf specification for which many software tools are readily available, it is still a fairly complex language with many subtleties and arcane features. On top of basic XML, a lot of other specifications have been built, not all of which are freely inter-operable that adds to the complexity. This means that the more all embracing XML processors and technologies are too complex for the straightforward needs of DDL particularly in lightweight systems. For these reasons a restricted subset of XML is defined for use in DDL.

4.2.2.1 Encoding and Detection of Encoding

DDL documents shall be encoded in Unicode encoding forms UTF-8 or UTF-16 [Unicode]. or in 7-bit US-ASCII [US-ASCII].

UTF-16 is indicated by the presence of a byte order mark that shall be present if this encoding is used. Since US-ASCII is a subset of UTF-8, in the absence of a byte order mark or explicit encoding declaration, a consumer of DDL that treats the document as UTF-8 will not encounter or generate any errors.

4.2.2.2 Restrictions

DDL shall not contain the following XML productions:

- PI
- doctypeddecl
- CDSect
- EntityRef – except the predefined entities
- Mixed content

4.2.2.3 XMLDecl

An XML declaration (XMLDecl) is not required. If an XML declaration is supplied it shall specify `version="1.1"`.

Encoding declarations if present must indicate the encoding used. Note though that in order to read the encoding declaration, encoding will have already been established so this is not relevant to parsers.

SDDDecl if present must indicate `standalone="yes"`

4.2.3 DTD, Schemas and Validation

In XML the syntax and grammar rules for a particular language or document format created from XML are called a schema. A document type declaration (DTD) as described in the XML 1.0 recommendation is a form of schema and a number of more advanced schema languages and forms have been published.

4.2.3.1 DDL Document Type Definition (DTD)

A DTD for DDL is given in Generic DTD DDL and schemas specific to DDL with DMP are also available. These are made available as an aid to creation and parsing DDL using standard tools but are non-normative. Because further rules are also given in the text and because the capabilities of schema languages vary, validation against a DTD or particular schema does not guarantee full conformance to the DDL syntax.

4.2.3.2 Use of Validation

An author creating DDL documents by whatever method should validate DDL documents when they are created. The author must be aware that validation against a formal schema does not guarantee that the document is correct or even that it makes sense.

Interpreters of DDL documents (controllers etc.) are not normally expected to validate DDL documents. Usually they will be parsing XML for specialist purposes that go much deeper than formal validation anyway.

4.2.3.3 Embedding Schemas in Documents

A DDL document shall not contain a Document Type Declaration (XML production: doctypeddecl) either internal or external. Nor shall it contain any other embedded schema. Links with DTDs or schemas for validation purposes must be made externally.

4.2.4 Namespace

This DDL version is identified with the namespace URL: “<http://www.esta.org/acn/namespace/ddl/2008/>”. See [XMLnames].

In common usage of DDL within control systems documents are homogeneous DDL and have no requirement for namespaces. However, the namespace above is provided for use when required.

Compliant controllers must be capable of recognizing namespace declarations and identifying the namespace of elements and attributes, but are not required to process nodes from any other namespace – such nodes may simply be discarded.

If namespace declarations are present, the DDL namespace shall be the default namespace within any <DDL> element.

4.3 Attribute and Element Value Types – XSD

XML attributes and elements with text content can represent names, numbers, dates and a host of other things. To standardize the text representation of such things, DDL uses the data type representations of XML Schema Part 2: Datatypes [XSD]. The prefix “xsd:” is used throughout this specification to denote such a type e.g. “xsd:nonPositiveInteger”, “xsd:NCName”. Please refer to XML Schema Part 2: Datatypes. for further information and full specification.

4.4 DDL Modules – Devices, Behaviorsets and Languagesets

Three elements in DDL descriptions are defined as “modules”. These are [device](#), [behaviorset](#) and [languageset](#).

Each module shall be assigned an identifier that is unique to that definition. This identifier may be used as a key to recognize already known modules or to retrieve a document containing the definition for a specific module.

A module identifier shall be a 128-bit universally unique identifier (UUID). The algorithm used to generate these is specified in [UUID]. This choice of algorithm means that an author of device descriptions can generate module identifiers without explicit recourse to an authorizing body. It also means that where required, a computer can generate its own identifiers (e.g. if there is a requirement to generate DDL on the fly).

Each DDL document (as defined by the XML document definition) shall contain exactly one module.

4.4.1 Module Content May not Change

All copies of a DDL module definition as identified by a given UUID, shall have exactly the same information content no matter where or how they are stored, or retrieved. This means that copies may vary in such things as encoding (subject to encoding rules see: Encoding and Detection of Encoding), insignificant white space, order of attributes etc. but the result of parsing any copy of the module, however transferred must always be the same.

For purposes of this rule XML comments shall not be regarded as part of the information content.

If any change is made to the information content of a module, then a new UUID shall be assigned and a new module is created.

4.4.2 Module Versions and Variations – [alternatefor](#)

If any change is made to the information content of a module, then a new UUID must be assigned. There is no explicit version mechanism provided in DDL and changes or improvements to modules may be made by different bodies from the original [provider](#).

However, a DDL module that is intended to replace or update an existing module (e.g. for corrections, addition of extra languages, minor upgrades, etc.) may indicate so using [alternatefor](#).

Authors using [alternatefor](#) must ensure that the new module they are adding it to genuinely covers all instances of devices conforming to the module it is replacing. They must consider that the old module may have been re-used by other users (as happens with low level modules) or may be used in the future without consideration of the newer version.

The interpretation of [alternatefor](#) by processors is advisory only. [alternatefor](#) suggests replacement and in

many cases the replacement will be an improvement, however, since there is no process for registering, certifying or policing descriptions this cannot be guaranteed in all cases.

4.4.3 Module Extension – [extends](#)

Another case of variation on a module occurs when the original is completely unchanged but new features are added – the new module is a superset of the original. A module can indicate this using [extends](#) that identifies the module upon which the current one is a superset. A processor can then recognize the underlying module even though the extended one has a new UUID.

The rules and semantics of [extends](#) depend on the type of module, as explained in clauses 4.4.3.1 through 4.4.3.3.

4.4.3.1 languageset

A languageset may be extended by adding versions of existing strings in languages in which they were previously undeclared, by adding completely new languages, or by adding new strings. Existing [alternatefor](#) declarations shall remain unchanged but new ones may be added where new languages are added. All the strings of the original languageset shall be present with unchanged keys and text in all the same language versions.

4.4.3.2 behaviorset

A behaviorset may be extended by adding new behavior definitions. Provided all the behavior definitions of the original behaviorset are present unchanged then the new behaviorset is an extension of the original.

4.4.3.3 device

For a new device to extend an original, the entire property tree of the original shall be present with identical functionality in the new device. New properties may be added provided they do not in their default states cause different behavior from the original device in any way. A controller must be able to access and control the new device exactly as though it were the original type and achieve the same functionality as the original would have provided.

4.5 Root Element

The root element of any DDL document shall be [DDL](#). This element shall contain exactly one module (see: DDL Modules – Devices, Behaviorsets and Languagesets).

4.6 Warning: Documents, Files and Resources

The XML syntax defines a document that is a logical structure. A document as defined by XML does not necessarily correspond with a file as defined by a computer filesystem or with a single resource as referenced by a URI. There may be applications or environments where it is necessary to combine multiple resources or files to create a document, or to separate a single file or resource into multiple documents. These cases are beyond the scope of this specification but must be clearly defined where they occur.

4.7 String Resources

As part of documenting devices, DDL provides a mechanism for collections of string resources to be stored in multiple alternative languages within DDL documents. These resources are available for labeling parts of the description, but are also available to devices themselves to call by reference as property values, either directly or indirectly via mechanisms such as selectors (referencing of strings from property values depends on appropriately defined [behaviors](#)). This means, for example, that a property that returns a numeric error code can (and should) be described in such a way that a controller can report verbose error messages in

multiple languages in response to those error codes.

Each [string](#) associates a text – the value of the string, which is intended for presentation to the user or other human readers – with a [key](#) that is a short name that identifies the string by lexical matching.

Strings are contained in language sets that define a set of keys and their replacement texts in various languages. There shall be no more than one text value per key, per language in a languageset.

4.7.1 Languages and Search Order

A [languageset](#) element contains one or more [language](#) elements. Each [language](#) element contains a set of strings in the (human) language specified by the [lang](#) attribute of the [language](#) element.

Within a [language](#) element each string key shall be unique. However, the same key may appear in multiple [language](#) elements, and the replacement text will therefore be in the language defined by the [language](#) element in which the string occurs.

Example 5: example of languageset definition

```
<languageset ...>
  <language lang="en-us">
    <string key="colr">color</string>
  </language>
  <language lang="en-gb">
    <string key="colr">colour</string>
  </language>
  <language lang="fr">
    <string key="colr">couleur</string>
  </language>
</languageset>
```

In this example three alternative language translations corresponding to the key “colr” are provided: American English, British English, and French.

[language](#) elements are not required to include text for all keys present in the [languageset](#), but a [language](#) element that does not include strings for every [key](#) shall specify an alternative language to substitute if a value for a key is not present. Thus given a [key](#) and a preferred language, a processor first searches the [language](#) element corresponding to the preferred language for a [string](#), and if one is not found repeats the process for the specified alternate language, and so on recursively until a value is found.

Example 6: languageset definition with incomplete languages

```
<languageset UUID="9fcb433e-8dd4-4418-9559-cb1a8e51648a">
  <language lang="en-us">
    <string key="colr">color</string>
    <string key="tim">time</string>
  </language>
  <language lang="en-gb" altlang="en-us">
    <string key="colr">colour</string>
  </language>
  <language lang="fr">
    <string key="colr">couleur</string>
    <string key="tim">temps</string>
  </language>
</languageset>
```

Here the British English language element does not include all strings but specifies that for strings not

present, the American English language element should be used.

The chain of alternate languages shall produce a text value for any key that is present in the languageset irrespective of which language is first chosen (out of those represented anywhere in the languageset). Any failure of this rule will either result in a dead-end at a language that does not specify any further alternative, or will result in a loop of alternative languages that repeats itself. Both conditions are errors.

The specification of an alternate language is optional, but if a [language](#) element does not specify an alternate then it must contain values for every key in the set.

One easy way to ensure that any key generates a value without recursive loops in the search is to ensure that there are one or more languages within the set that contain values for all keys in the set, and that all chains of alternate languages terminate at one such language.

4.7.2 Identification of Strings

Following the rules above, to unambiguously refer to a string text, three items are required:

- The languageset containing the string
- The string's key
- The preferred language (in the form of a language tag [LANG-TAG]).

The first two items must be provided by the description or by the device itself anywhere a string reference is used. The preferred language is normally a matter for the processor that is presenting the values and is typically a user configuration choice.

Example 7: Labels Using String References

```
<device>
  <UUIDname UUID="9fcb433e-8dd4-4418-9559-cb1a8e51648a"
    name="exampleStrings"/>
  ...
  <property ...>
    <label set="exampleStrings" key="colr">
      ...
    </property>
  <property ...>
    <label set="exampleStrings" key="tim">
      ...
    </property>
</device>
```

The property labels above use the strings in Example 6: . With preferred language set to “fr” the two property labels are “couleur” and “temps”, with language “en-us” they are “color” and “time” and with language “en-gb” they are “colour” and “time”.

4.8 Arrays of Properties

Arrays of properties of the same type or function are common in devices and can range from simple array values (e.g. the levels of a graphic equalizer or an arbitrary dimmer curve), to arrays of complete sub-trees or sub-devices (e.g. an automation rack with an array of motor controllers).

In order to avoid repetitive text and to highlight to controllers where arrays of identical properties occur, the syntax allows any property or included sub device to be declared as an array. This means that not only that property or sub device but the entire sub-tree of which it is the root is actually an array of identical sub-trees.

The mechanism for defining the different addresses of network values within an array is protocol dependent and must be specified as a part of the access protocol interface specification.

Where properties within an array are declared with immediate values, then an array of values must be provided.

4.8.1 Multidimensional Arrays

A property with sub-properties forms a branch of the tree and if declared as an array, it forms an array of branches. If one among its sub-properties is also an array, then an array of arrays or multidimensional array is declared. Thus while the specification of individual properties only allows array declarations in one dimension, it is quite possible to generate multidimensional arrays by nesting arrays within arrays.

In order to avoid ambiguities, applications must keep track of the count of each dimension of an array individually – that is, an array of m properties with a child array of n properties must be treated as a two dimensional array $[0 \dots m-1][0 \dots n-1]$ and not simply as an array of $m \times n$ properties $[0 \dots m \times n-1]$.

4.8.2 Properties That Do Not Iterate Across an Array

In some cases of arrays, it is required for some properties in the sub-tree not to iterate with the array. For example, an array of scalar variables may share a common maximum limit. This is an application for the generic shared property mechanism for representing properties with multiple parents and in the simplest case, there is a special simplified mechanism (see: Shared Properties).

4.8.3 Immediate Values for Array Properties

If an array property has an immediate value, then in general a value must be provided for each property of the array. However, the case where all elements of the array have the same immediate value is so common that a special case is made and it is not necessary to use the shared property mechanism.

In general for any array property with an immediate value, the description shall provide either one single value that applies to all elements of the array, or a number of values equal to the maximum number of properties in the array. For example, for a property with array size $[a][b][c]$ then there shall either be 1 or $a \times b \times c$ values. That is, if there are multiple values, then these must be given separately for each element within each dimension of the array.

For multidimensional arrays property values shall be provided in inner array first order. For example for a property array size 3 containing a child property array size 2 the values shall be in order $V_{0,0}$ $V_{0,1}$ $V_{1,0}$ $V_{1,1}$ $V_{2,0}$ $V_{2,1}$

4.8.4 Parametric Array Sizes

When the size of an array is determined by a parameter value (see: Parametric Devices), then it is not always possible to specify exactly the right number of values for each property of the array. However, rules governing parametrization of array sizes require that a maximum size is always known and this is the number of values that shall be supplied. It is therefore not an error for there to be more values provided than the actual size of the array determined by the parameter value.

In the case of multi-dimensional arrays, where the size in one or more dimensions is supplied by a parameter, a DDL processor must use the maximum value for the array size rather than the supplied parameter, in calculating how to iterate between higher dimensions in the array.

4.9 Element Identifiers

The XML specification defines the use of two types of attributes, ID and IDREF, as a mechanism of cross-

linking between XML elements. This mechanism is extensively used and implemented by XML processors for a wide variety of purposes. A very brief specification [xml-id] outlines the use of the attribute name “xml:id” for ID type attributes so that they can be easily recognized in the absence of specific schema information.

Nearly all elements within DDL may carry an xml:id attribute that shall conform to the [xml-id] spec. As well as use within generic XML tools (it is useful in Xinclude [XInclude], Xpointer [XPointer], Xpath [XPath] and many higher applications), DDL uses and extends xml:id for structural references to the items represented by the device model. However, this use is constrained where references are limited within a single document so DDL extends this to allow references across devices.

4.10 Use of XML Identifiers to Reference Device Model Features

The use of [xml:id](#) attributes on properties, or other parts of the device model provides a mechanism to build cross references between parts of devices. In order to do this, any reference must identify an instance of the device. The match then identifies by way of the DDL, the property or other item represented by the element within the device that is referenced. Finally, since in the device model a match might identify a whole array of items (represented by just one element in the actual XML), rules for referencing across arrays may need to be applied.

References of this sort are used directly by the shared property mechanism. They are also used extensively by a number of behaviors that rely on pointers or references between properties.

4.10.1 Reference to Elements in Device Trees – Extended Reference Syntax

The extended reference syntax is specific to references to instances of devices. It is used by the shared property mechanism (shared attribute) and may be used by behaviors or future specifications.

Bearing in mind that each DDL device is a separate module and therefore a discrete and complete XML document.

- The syntax defines an extended reference using a series of steps.
- Successive steps shall be separated by the character “/”.
- For each step there is a context device. Each step except the last shall identify a device relative to the context device that shall then be used as the context device for the subsequent step.
- If the reference begins with /, the context device shall be a root device in an appliance. If there is no explicitly indicated appliance (e.g. by a separate reference to a network component) this shall be the appliance that contains the reference.
- If the reference does not begin “/”, the context device for the first step shall be the device containing the reference.
- A step whose text is “.” identifies the context device itself.
- A step whose text is “..” identifies the parent device of the context device.
- A step whose text is anything else shall be an XML Name, which may optionally be followed by array subscripts and/or an array matching specifier – see below. The Name is an [xml:id](#) reference within the context device and matches the [xml:id](#) attribute of an element within that device. e.g. step “foo” identifies the element with `xml:id="foo"`.
- A step that identifies an includeddev element and that is not the last step, establishes the device instance specified by the includeddev as the context device for subsequent steps (see: Statically Included Device Descriptions).
- A step that identifies an property element that has device reference behavior and that is not the last step, establishes the device instance specified by the device reference as the context device for subsequent steps (see: Sub Devices Attached by Reference).

- Any step that is not the last and that does not identify either an includeddev, or device reference property is an error.
- The last step may identify any element within the context device established by all preceding steps.
- *Array Subscripts*: A reference that identifies an element that is a part of a property or includeddev array, represents multiple items in the description. The specific item may be indicated by subscripting the [xml:id](#) reference step using square brackets. If subscripting is used, there shall be exactly one subscript for dimension of the array within the context device. Multiple subscripts shall occur in the same order that the array elements are nested – outermost first. e.g. `ref="foo[3][201]"`.
- A step that identifies an element that is a part of a property array and for which no subscript is specified is subject to the rules in section References in Arrays.
- A reference that is itself in an array (by being declared as a descendent of an array property) is an array of references and shall resolve according to the rules in section References in Arrays.
- *Array Matching Specifier*: When a reference is made from within an array the last step of the reference may be an Array Matching String enclosed in either single or double quotes. The Array Matching Specifier shall follow any subscripts provided in the same step.

4.10.1.1 Examples

`ref="localprop"` references an element with `xml:id="localprop"` in the current device.

`ref="mysubdev/elemsubdev"` references element with `xml:id="elemsubdev"` in the device whose `xml:id="mysubdev"` in the current device.

`ref="/foo[8][12]/bar[5]/boff"` references element with `xml:id="boff"` in subdevice 5 in the array of subdevices with `xml:id="bar"` in subdevice number 12 in sub-array number 8 in the array of subdevices with `xml:id="foo"` in the root device of the appliance.

4.10.2 References in Arrays

In DDL both a reference and/or its target may occur within an array. This creates four types of reference: one-to-one; one-to-many; many-to-one and many-to-many. propertypointer references cannot be one-to-many as this creates ambiguity. The rules provided below apply to propertypointer and any property-reference behaviors in which one-to-many references are forbidden and ensure that references can be unambiguously resolved.

A reference to an array that specifies a subscript is pointing to a singular item. In the general case of multidimensional arrays, a subscript in the reference may still identify a sub-array of targets so for each subscript provided the number of dimensions of the target array is reduced by one.

In the multidimensional case, taking a single dimension at a time, there are two ways to match an array of references to the target: correspondence match and many-to-one match.

4.10.2.1 Single Dimension Correspondence Match

The array size of the target item must be identical to the array size of the reference. Each reference shall be interpreted as pointing to the item at the same position in the target array as the position of the reference in its own array (1:1 correspondence).

4.10.2.2 Single Dimension Many-to-one Match

The target is a single item. Each reference points to the same item.

4.10.2.3 Multidimensional Arrays

In the case of multi-dimensional arrays, the same reference types are generalized, with each array being an array of items, and each item being a sub-array, except the innermost case where items are singular.

When a reference is made within an R dimensional array, and the target element is within an T dimensional array. The following rules are applied recursively until all dimensions are matched:

- If $R < T$ it is an error and no match can be made.
- If $R = T$ then matching shall be by multidimensional correspondence match as described below.
- If $R > T$ then matching shall be either by multidimensional correspondence match, or by multidimensional many-to-one match as described below.

4.10.2.3.1. multidimensional correspondence match

The size (the array item count) of the outermost dimension of the reference array shall be exactly equal to the size of the outermost dimension of the target array. Each item in the reference array points to the corresponding item in the target array. Each reference item is now a $R-1$ dimensional array of references pointing to a $T-1$ dimensional array of targets, for which these rules are applied again.

4.10.2.3.2. multidimensional many-to-one match

All members of the outermost dimension of the reference array, point to the same target array. There remain $R-1$ dimensions of reference, pointing to a target array that still has T dimensions for which these rules are applied again.

4.10.2.3.3. Notes

It follows from the rules above, that in order to make matches between multidimensional arrays of references and targets:

- the number of dimensions of reference must be greater than or equal to the number of dimensions of target ($R \geq T$);
- for each dimension of target, there must be a correspondence match with a single dimension of reference in which the array sizes must be the same;
- any excess dimensions of reference must be many-to-one matches;
- the ordering of dimensions that have correspondence matches must be the same in both arrays.

4.10.2.4 Array Matching String syntax

An array matching string specifies how the dimensions of an array of references are to be matched to an array of targets:

- Each character in the string shall correspond to a single dimension of the reference array with the first (leftmost) character representing the outermost array dimension, and the last representing the innermost.
- Each character shall be either “=” or “*”.
- A dimension whose character is “=” shall be correspondence matched with the target array according to the rules above.
- A dimension whose character is “*” shall be many-to-one matched with the target array according to the rules above.

It follows from this, that where there are R dimensions of reference to T dimensions of target the string must have R characters of which T must be “=” and the rest “*”.

4.10.2.5 Default Matching for Multi-dimensional Arrays

In the absence of explicit specification (e.g. by supplying an array matching string), a multidimensional match from an R dimensional array of references to a T dimensional array of targets shall be performed by correspondence matching the first (outermost) T dimensions and many-to-one matching any remaining dimensions.

4.10.2.6 Examples

These examples all use propertypointer references to illustrate matching – see below.

The following XML creates a 3×4 array of target properties named $Tgt_{i,j}$ that can be visualized like this:

```

    Tgt0,0 Tgt0,1 Tgt0,2 Tgt0,3
    Tgt1,0 Tgt1,1 Tgt1,2 Tgt1,3
    Tgt2,0 Tgt2,1 Tgt2,2 Tgt2,3
<property array="3">
  <property array="4">
    <property xml:id="Tgt" sharedefine="true">
      ...
    </property>
    ...
  </property>
  ...
</property>

```

Example 8: One-to-one Match

This is a one-to-one matching array of references named $Ref_{i,j}$:

```

    Ref0,0 → Tgt0,0, Ref0,1 → Tgt0,1 ... Ref0,3 → Tgt0,3
    ...
    Ref2,0 → Tgt2,0 ... Ref2,2 → Tgt2,2, Ref2,3 → Tgt2,3
<property array="3">
  <property array="4">
    <propertypointer ref="Tgt"/>
  </property>
</property>

```

Example 9: Mixed one-to-one and many-to-one matches

This is a 3 dimensional array of references in which the second dimension is many-to-one matched:

```

    [Ref0,0,0...Ref0,99,0] → Tgt0,0, [Ref0,0,1...Ref0,99,1] → Tgt0,1 ... [Ref2,0,3...Ref2,99,3] → Tgt2,3
<property array="3">
  <property array="100">
    <property array="4">
      <propertypointer ref="Tgt'*= '"/>
    </property>
  </property>
</property>

```

4.11 Shared Properties

Section 3.4.1.2 introduced the idea of shared properties that represent places where branches of the property tree recombine so a property has multiple parents. A shared property in DDL is declared in multiple places. Exactly one declaration shall be a defining-declaration. All other declarations of the shared property shall be reference declarations that are called propertypointers.

4.11.1 Defining-declaration

The defining-declaration of a shared property is a normal property declaration in which the full structure of the property including all its behaviors, its value, its sub-properties etc. appear. It is identified as a shared property by setting its [sharedefine](#) attribute to “true” or to “arraycommon”. Since it will be referenced using the reference syntax above, it must also carry an [xml:id](#) attribute.

It is not an error if a property with `sharedefine="true"` is not referenced. It may be referenced from outside the device and the presence of such references may depend on how this device relates to other devices in a particular appliance.

4.11.2 propertypointer

A *propertypointer* contains a [ref](#) attribute whose value is a reference that obeys the syntax rules for extended references and references in arrays (see: Reference to Elements in Device Trees – Extended Reference Syntax, References in Arrays). It has no behaviors, child elements or other structure – these must be declared at the defining-declaration – but in interpreting the DDL, the *propertypointer* represents the same property as the defining property to which it points in each place it is declared, so by definition all its behaviors, values and sub-properties are the same.

The rule in Section Branch Re-combination and Shared Properties forbidding any property to be simultaneously both parent and child of any other, or of itself means that a *propertypointer* cannot resolve to itself or to any of its ancestor properties whether directly, or indirectly via any number of levels of references or parent or child devices.

It is permitted for a *propertypointer* to reference another *propertypointer* (they both represent the same property) provided that such a chain of references resolves to a full property declaration eventually. The target of the [ref](#) attribute of a *propertypointer* shall be a property whose [sharedefine](#) attribute is set to either “true” or “arraycommon”.

It is not an error if a *propertypointer* does not resolve when considering a device description in isolation, but It is an error if any instance of a *propertypointer*, does not resolve unambiguously to a defining-declaration of the shared property when the device is built, together with other devices, into an appliance (and after any parameter substitutions have been made. See: Parametric Devices). Note that while the [xml:id](#) of a defining declaration cannot be parametrized, it is allowable to parametrize [ref](#) attributes to ensure that *propertypointers* within template devices resolve correctly.

4.11.3 Special Case – Property Shared Across an Array

The case that a common sub-property exists within an array that is the same property for all branches of the array occurs sufficiently often that a special-case syntax is available to improve readability.

Any property within an array (of any number of dimensions) whose [sharedefine](#) attribute is set to *arraycommon* shall be interpreted as a single property that occurs in all branches of the array.

The case where a property with attribute `sharedefine="arraycommon"` is not in an array is the special case where the array size is 1 and is not an error.

The case of a property that has both attribute `sharedefine="arraycommon"` and attribute `array="N"` where $N > 1$, is forbidden.

A property with `sharedefine="arraycommon"` may be referenced with a [propertypointer](#) in the same way as for `sharedefine="true"`. As a single property, any references to it from within an array must follow many-to-one matching rules.

Note

The very common case where a sub-property within an array has an immediate value and no children does not require `shareddefine="arraycommon"` because a single value is automatically applied to all items in the array.

4.12 Declaration of of Sub Devices

As mentioned in Hierarchical Device Structure, sub devices in DDL may occur anywhere that one or more properties are legal and become a logical part of the entire device property tree. DDL provides two mechanisms for declaring sub devices that have different applications.

4.12.1 Statically Included Device Descriptions

Any device description (identified by its UUID) may be included within another description at any point that a property is allowed using an [includedev](#) element. When included in this way, all the properties of the included device become part of the tree of the description that includes them, at the point they are included.

Devices included by static inclusion are entirely specified within the text of the description and so because of the rules preventing changes (see: Module Content May not Change), cannot be used for dynamically variable sub devices (e.g. interchangeable modules). For example, if description of A includes device B, then B becomes a fixed part of A.

Static inclusion does however, provide the opportunity to specify parameters to the included device and this mechanism allows a template description to be included with different parameters in different places (see: Parametric Devices).

[Access protocol](#) specific information may also be specified at the point of inclusion depending on the rules of the specific access protocol(s).

Example 10: Static Inclusion Sample DDL

```
<DDL version="1.1">
  <!-- First we define a color wheel device -->
  <device UUID="c158b08d-e03e-43b3-88b2-d56dc1447155" ...>
    <label>Color Wheel</label>
    <property valuetype="network" ...>
      <label>color selector</label>
      ...
    </property>
  </device>
  <!-- Now we can use the definition -->
  <device UUID="77ee2876-ed3a-4728-b789-2a330d55c051" ...>
    <property ...>
      <label>color changing luminaire</label>
      ...
      <includedev UUID="c158b08d-e03e-43b3-88b2-d56dc1447155"/>
      ...
    </property>
  </device>
</DDL>
```

4.12.2 Sub Devices Attached by Reference

The second method uses a property to mark an attachment point where a sub device is attached to the

description tree. This property is called a device reference property and shall be identified by a suitably defined behavior. The device reference behavior must be defined in the basic set of behaviors for any protocol adaptation in which this method of attachment is to be used. The value of a device reference property is the UUID of the device attached. So far as the control model is concerned, the property tree(s) of the sub device can be considered to replace the property that marks its attachment.

Because the device attached in this way is specified by a property value, it follows all the usual rules for property values and may be specified as an immediate value embedded within the description or a network value that is accessed over the network. If a network value, it can be constant or may vary – for instance as hot-plug modules are added or removed or as a software defined device such as an Audio DSP is reconfigured. The UUID may also be a driven value – for example driven by a selector property to select a value from a set of choices, thus allowing network control over the choice of sub-device.

Attachment of sub devices by reference does not however, allow parameters to be directly specified for the attached device. (see: Parametric Devices).

Example 11: Device Reference Sample DDL

```
<DDL version="1.1">
  <!--
    Define a color wheel device
    (same description as previous example)
  -->
  <device UUID="c158b08d-e03e-43b3-88b2-d56dc1447155" ...>
    <label>Color Wheel</label>
    <property valuetype="network" ...>
      <label>color selector</label>
      ...
    </property>
  </device>
  <!-- This luminaire has an optional color wheel -->
  <device UUID="77ee2876-ed3a-4728-b789-2a330d55c051" ...>
    <property ...>
      <label>color changing luminaire</label>
      ...
      <property valuetype="network">
        <behavior set="example" name="device_reference"/>
        <!--
          if the color wheel is fitted
          this property will hold the value:
            c158b08d-e03e-43b3-88b2-d56dc1447155
          otherwise it will hold NULL indicating no sub-device
        -->
        ...
      </property>
    </property>
  </device>
</DDL>
```

As shown in the example, a device reference property whose value is the null UUID (all zeros) shall indicate that no subdevice is present.

4.12.3 Circular References

Whenever a the UUID of a subdevice is declared within a description, whether by static inclusion or using a

device reference property with an immediate value, there is no way that the UUID can change from instance to instance – it is invariant. It is an error for any device description to reference its own UUID as an invariant sub device or, recursively as any invariant descendant (a class cannot be its own parent or ancestor).

It is also an error for any instance of a device to reference itself as a sub device or other descendant.

4.12.4 Which Method to Use?

If a child device needs to be dynamically attached or to vary from instance to instance of the parent then attachment by reference is the only method that can work.

If any parameters of the included device are to be varied then static inclusion can specify these directly and should be used.

If a child device is statically defined and constant for all instances of the parent, and included purely to gain benefits of device re-use, then either static inclusion or attachment using a immediate valued reference property may be used.

4.12.5 Discovery of Device Structure

The mechanism for discovery of a root level device for a piece of equipment (in ACN protocols this is a [component](#)) is protocol dependent. However, the [DCID](#) is a key that may be used to retrieve the XML text of the definition itself. Within a definition, references to subsidiary devices are made by [DCID](#) that then identifies the XML text of their definitions. Thus once a root level device has been identified the entire tree can be constructed, provided that all necessary device descriptions are available.

4.13 Parametric Devices

The rule that module content may not change (see: Parameter Substitution) means that a processor that encounters and recognizes a device class it already knows, can be confident that the description is no different from one it has already parsed. However, there are many examples of devices that have the same control structure but differ in a few parameters. For example many cars share the same control structure of steering wheel, gearstick, clutch accelerator and brake, but differ in top speed, turning circle, gear ratios and so on. It would be possible to describe all such cars using a fixed device class while using network property values for values that differ, but this means that a controller needs to have the actual instance available on line before it can configure itself for specific values. In entertainment technology and any other applications where off-line configuration is commonplace, this is a serious disadvantage.

If, on the other hand, a separate device is declared for each variant, this allows the top speed and other items to be specified within the description as immediate values but because each device has a separately identified description the commonality of structure is lost.

DDL's solution is to allow individual items in the description to be marked as parameters. These are called *parameter fields*. Any device may declare zero or more named parameters, each of which applies to one or more parameter fields.

When a device that contains parameters is statically included in a description (using [includedev](#)), values for any of the parameters of that device may be defined. The included device is called a template device.

4.13.1 Parameter Fields

A parameter field shall be either an attribute value or the text content of an element (#PCDATA in DTD terms). In either case the parameter value is a text string and cannot contain XML elements or other markup. Each parameter field is bound to a single parameter name (see below). A parameter name may have multiple fields bound to it.

4.13.2 Defining Parameter Names

In order to define a parameter in a device, one or more [parameter](#) elements shall be provided near the top of the device definition. Each parameter element provides a name for the parameter, and may optionally specify various restrictions on the value that can be supplied for that parameter (see: Restricting Parameter Values).

4.13.3 Parameter Substitution

When a template device is included within a parent device using [includedev](#), each of its parameters may have a string value provided (using a [setparam](#) element), and this value shall then be used in place of whatever value that was present in the original field(s) when interpreting the DDL of the template device.

Any value supplied in a [setparam](#) element shall be valid and legal in all fields that are bound to that parameter and shall meet all restrictions imposed in any [parameter](#) declarations declaring that parameter.

4.13.4 Restricting Parameter Values

When a parameter is declared, the range of values it is permitted to take may be left unbounded or may be restricted in several ways. If no restrictions are specified, then any value that is legal for all fields bound to that parameter may be specified for the parameter when the device is included. If a restriction is imposed then it is illegal to specify a parameter value that contravenes the restriction when including the device. Restrictions are defined by [choice](#), [mininclusive](#), [maxinclusive](#) or [refinement](#) elements.

4.13.5 Binding Fields to Parameters

For any field within a device that can be parametrized, there is a corresponding parameter-binding attribute, whose value is the name of the parameter to which that field is bound. Since this attribute contains a single name, it is impossible to bind a field to more than one parameter. It is an error if the name does not match one specified in a [parameter](#) element within the same device.

Each parameter-binding attribute consists of the name of the field (element or attribute name) with the suffix “.paramname” appended. For example, a property array size is specified with the array attribute. The corresponding parameter binding attribute is *array.paramname*, so to bind the array size to parameter “num-props”, you might specify `<property array="10" array.paramname="num-props" ...>`. To bind a label with literal content (the parametrized field is the element content, not an attribute) you might write `<label label.paramname="labeltext">...</label>`.

Only those fields for which a corresponding [.paramname](#) attribute is explicitly declared in the rules of Section 5 can be parametrized.

One field that shall never be parametrized is the [xml:id](#) attribute because parametrizing this would break interoperability with many standard XML processing applications.

Future Extensions to DDL

This mechanism means that if any attribute of DDL has the same name as its parent element then neither the attribute, nor the parent element can be made a parameter field without ambiguity. This does not create any problems with current DDL that has no instances of attributes having the same name as their parent, but must be considered when extending DDL (e.g. for new protocols).

When creating new elements and attributes for DDL (e.g. for new protocols), it should be the rule to make all fields parametrizable, by creating corresponding [.paramname](#) attributes, unless doing so would create a specific problem or conflict. It is unacceptable not to create a [.paramname](#) attribute simply because there is no obvious reason why a user would want to parametrize the field.

Because parameters are only in scope in the body of a device and after the point of declaration, there are some elements whose attributes can never be parametrized. These include [DDL](#), [device](#), [UUIDname](#), [behaviorset](#), [languageset](#) and all child elements of [behaviorset](#) and [languageset](#).

4.13.6 Default Values

All of the requirements applying to fields within devices (as specified elsewhere in this Standard) apply whether or not that field is parametrized. Thus any parametrized field must have a valid value where it occurs in the description. If no value for the parameter is supplied when the template device is instantiated, or if the template device is attached by reference, or if it forms the root device of an appliance, then the value used for each parametrized field shall default to that given in place.

It is legal for differing default values to be given in-place for multiple fields of the same parameter, but there is no way to override these with differing values when that parameter's value is explicitly specified on inclusion using setparam. For this reason, supplying differing in-place values for different fields of the same parameter is confusing and rarely a good idea.

4.13.7 Attributes with Default Values

Many DDL attributes have default values specified (e.g. array defaults to 1) in the case they are not explicitly provided. To aid readability and avoid confusion it is strongly recommended that when binding such attributes to a parameter, the attribute itself as well as its .paramname attribute should be explicitly written in the DDL, even if its in-place value is the default.

In the case that such an attribute is not explicitly written, the DDL specified default value for the attribute shall be treated as the parameter default.

4.13.8 Template Example

The DDL shows fragments of DDL from a color-wheel template device whose color count can be modified when included.

Example 12: Static Inclusion With Parameters Sample DDL

```
<DDL version="1.1">
  <!-- First we define a parametric color wheel device -->
  <device UUID="c158b08d-e03e-43b3-88b2-d56dc1447155" ...>
    <parameter name="colorCount">
      <mininclusive>2</mininclusive>
      <maxinclusive>100</maxinclusive>
    </parameter>
    <label>Color Wheel</label>
    <property valuetype="network" ...>
      <label>color selector</label>
      ...
      <property valuetype="immediate" ...>
        <label>number of colors</label>
        ...
        <value value.paramname="colorCount">6</value>
        ...
      </property>
    </property>
  </device>
</DDL>
<DDL version="1.1">
```



```

<!--
  Now we can use the definition with 10 colors instead of 6
-->
<device UUID="77ee2876-ed3a-4728-b789-2a330d55c051" ...>
  <property ...>
    <label>color changing luminaire</label>
    ...
    <includedev UUID="c158b08d-e03e-43b3-88b2-d56dc1447155">
      <setparam name="colorCount">10</setparam>
    </includedev>
    ...
  </property>
</device>
</DDL>

```

4.13.9 Adoption of Parameters from Included Devices by Parent Devices

As seen above, it is not required to specify values for every (or even any) parameter of a template when instantiating it. Any parameter of a template device whose value is not specified using [setparam](#) at inclusion, automatically becomes a parameter of the parent device. Its name gains the same scope as other parameters within the parent and may have values specified in the normal way if the parent is itself included in a grandparent device.

Because of this inheritance, it is possible that multiple parameter definitions using the same name are in scope at the same time. This can occur when the same sub-device is included multiple times, when different included sub-devices have parameters of the same name, or when a parent device defines a parameter of the same name as one used within an included sub-device.

Since parameter matching is performed solely by name, any supplied value for a parameter shall apply to every parameter field whose name matches and is in scope, wherever they are specified. This is not an error and can be used to advantage, but great care must be taken when template devices are included, to ensure that any names that are adopted, either do not conflict, or make sense when a common value is applied in all places.

Note

If a parameter value is fixed at inclusion using `setparam`, the name of that parameter has no further visibility elsewhere in the parent (or in any device that includes the parent) and any use of the same parameter name elsewhere in the parent has no effect on the included device.

4.13.10 Re-parametrizing Supplied Values

It is legal for the value provided by a [setparam](#) element to itself be parametrized (using [setparam.paramname](#)). This allows for parameters within included devices to be renamed within the parent, and for new default values to be supplied. This in turn allows naming conflicts caused by adoption of parameters (see above) to be resolved. It also allows parameters to be defined within the parent that apply to fields within both the parent and the included device.

In the following example, the same color-wheel as the preceding example, is included twice in a double color-wheel device in which the color counts for wheel 1 and wheel 2 can be separately specified but default to 7 and 9.

Example 13: Nested Static Inclusion Sample DDL

```

<DDL version="1.1">
  <device UUID="77ee2876-ed3a-4728-b789-2a330d55c051" ...>
    <parameter name="colorCount_1"/>
    <parameter name="colorCount_2"/>
    <label>double color wheel template</label>
    ...
    <includedev UUID="c158b08d-e03e-43b3-88b2-d56dc1447155">
      <setparam
        setparam.paramname="colorCount_1"
        name="colorCount">7</setparam>
      </includedev>
      ...
      <includedev UUID="c158b08d-e03e-43b3-88b2-d56dc1447155">
        <setparam
          setparam.paramname="colorCount_2"
          name="colorCount">9</setparam>
        </includedev>
        ...
      </device>
    </DDL>

```

4.13.11 Notes and Additional Rules

The format and content of any parametrized value as supplied in a setparam element shall be governed by the rules provided for the individual field or fields to which that parameter applies and shall be a valid value within all such fields. It is an error if a single parameter applies to multiple fields for which there is no possible common value that is permitted and valid in all those fields.

A parameter shall have a name (specified by the [parameter](#) element) that is unique among all the explicitly identified parameters of the same [device](#).

The scope of a parameter name shall be the [device](#) in which it is declared (extending from the point of its declaration to the closing </device> tag, excluding any subdevices (except in the case of adoption of parameters from children as defined above). The scope also extends to any device that statically includes this device according to the rules in Section 4.13.9 .

Careless parametrization of arbitrary fields or sharing of parameter names between parent and child devices may make a template device difficult or even impossible to use – a template that when instantiated cannot generate legal DDL or can only do so for a single specific parameter value is pointless.

4.13.12 Dynamic use of Parameteric Devices

There is no direct mechanism to specify parameters when a device is attached by reference or when a root device is declared. However, it is quite easy to create a wrapper device that gives the required parameter values around a template and attach that.

In the previous example of a car with steering, gearstick, accelerator clutch and brake, once we have a full description of one such car, we can define another car device that does not introduce any new properties but includes the fully defined car at its top level and at the same time specifies new values for the various parameters. We now have two car devices, each with its own [DCID](#), but a very brief inspection of the description of the second will reveal that it is merely a variant of the first.

4.14 Logical and Syntactic Property Structures

The DDL syntax, often gives rise to properties that are declared for syntactic or structural reasons but that are not logically a part of the device model structure at a given time. This generates the concept of logical and syntactic trees and of structural properties. The syntactic tree is the tree of properties as it occurs in the DDL document and is defined by the XML. The logical tree is the tree of properties that represent device control model at any given time, properties that govern the structure of the logical tree but that are not a part of it are called structural properties. The logical tree is not necessarily fixed but can change when the value of certain properties change (often those in the syntactic tree that are not present in the logical tree).

Example 1: Device Reference

A device reference property has a value that identifies a sub-device. So far as the logical device model is concerned, the root property or properties of that sub-device are inserted in place of the device reference property to form a tree incorporating both the parent and child devices. However, if the value of the device reference property is changed (representing attachment of a different sub-device) then the entire logical tree from the device reference point down may be different. The device reference property is a structural property.

Example 2: Alternative Property Sets

A rotating part may be controlled by position (sometimes called indexing) or may rotate continuously and be controlled by speed. In some applications both means are useful and a property is used to switch between two alternative sets of control properties (e.g. see DMP's propertySetSelector behavior). In this case, the primary parent property is the rotation angle and the switch property is syntactically its child while the target position (for positional control) and speed (for speed control) are both children of the switch that chooses which of the two is active. The target position and speed are therefore syntactically the grandchildren of the rotation angle but whichever is active is logically the child of rotation angle since that is how target position or speed work. The control method selector switch is a structural property.

5 DDL Element and Attribute Reference

5.1 alternatefor

empty element

A new version of a DDL module may indicate that it replaces a previous version using the “alternatefor” element. This provides a mechanism for corrections, addition of extra languages, improvements, etc.

Authors using this element must ensure that the new module they are adding it to, genuinely covers all instances of devices conforming to the module it is replacing. They must consider that the old module may have been re-used by other manufacturers (as happens with low level modules) or may be used in the future without consideration of the newer version.

The interpretation of alternatefor by processors is advisory only since there is no formal version mechanism for modules and an alternate version may be provided by any organization.

Attributes

- [UUID](#) (required) the UUID the current module is intended to replace.
- [UUID.paramname](#) (optional) binds the UUID attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Parents

- [device](#)

- [behaviorset](#)
- [languageset](#)

5.2 altlang

attribute

Identifies the language to search if a string matching a particular [key](#) is not found in the current [language](#) element.

The values of the attribute shall be language identifiers as defined by *Tags for the Identification of Languages* [LANG-TAG] or its successor.

Value

Shall conform to [xsd:language](#)

Parents

- [language](#)

5.3 array

attribute

Declares that its parent property or included sub device is an array of identical properties, or sub devices. The value of array shall be a non-zero positive integer.

If array is not present the parent property or includeddev is a single instance. All child properties of an array property or within an array sub device are part of the array (Section 4.8, “Arrays of Properties”).

Value. Shall conform to [xsd:unsignedInt](#) with minInclusive set to 1. Default if not present is 1.

Parameter restrictions

When array is made a parameter field the range of allowable values shall always be restricted using either [choice](#) or [maxinclusive](#). This allows a maximum array size to be determined.

Parents

- [property](#)
- [includeddev](#)

5.3.1 Arrays of Values

When a [property](#) that is declared to be an array contains immediate values, either directly or within sub properties, the number of values given depends on the maximum size of the array.

- one if all properties in the array have the same value.
- the same as the value of array if any values differ and array is not a parameter.
- the same as the maximum allowable value of [array](#) if any values differ and array is a [parameter](#).

See also Arrays of Properties.

5.4 behavior

empty element

Identifies a single behavior that applies to its parent property. This is a reference to a behavior definition that must be contained in a behaviorset. In order for the reference to be unambiguous, both the behaviorset and the name of the behavior within that set must be given.

Attributes

- [name](#) (required) the name of the behavior identified.
- [name.paramname](#) (optional) binds the name attribute to the named parameter.
- [set](#) (required) the UUID of the behaviorset within which the behavior definition occurs.
- [set.paramname](#) (optional) binds the set attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Parents

- [property](#)

5.5 behaviordef

element

Contains the definition of a property behavior that is specified in one or more sections. This specifies the behavior of any property that references it.

Each behaviordef has a name that must be unique among all behaviordef elements within its containing [behaviorset](#). For readability, the name should provide a meaningful reminder of the behavior it identifies.

Each behavior should be a refinement of one or more, more generic or abstract behaviors (see: Behavior Derivation and Refinement, [refines](#)).

The content of the behavior element is a text description of the semantics of the behavior. To allow structuring of the prose contained within behaviordef, it is divided into [sections](#) – that may be nested. At least one [section](#) is required. Each section has an optional [heading](#) and may contain text [paragraphs](#) and subsections. Processors presenting behaviors to users should use these to assist in layout. (See: Behavior Definitions).

Attributes

- [name](#) (required) must be unique within the containing [behaviorset](#) module.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children (in order)

1. [label](#) (zero or one)
2. [refines](#) (zero or more)
3. [section](#) (one or more)

Parents

- [behaviorset](#)

5.6 behaviorset

element

A behaviorset contains a set of named behaviors. Each behavior definition is referenced by the set in which it occurs (identified by its UUID) and by its name within that set.

Attributes

- [UUID](#) (required) the unique identifier for this behaviorset
- [provider](#) (required) the organization that published this behaviorset
- [date](#) (required) the publication date of this behaviorset
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children (in order)

1. [UUIDname](#) (zero or more)
2. [label](#) (zero or one)
3. [alternatefor](#) (zero or more)
4. [extends](#) (zero or more)
5. [behaviordef](#) (one or more)

Parents

- [DDL](#)

5.7 choice

element

Specifies one of a set of permitted values for a parameter (see: Statically Included Device Descriptions, Parametric Devices). Each value within the set is specified by a separate choice element.

If a choice element is present within a [parameter](#) element then no [mininclusive](#), [maxinclusive](#) or [refinement](#) element shall be present in the same parameter.

The value given by a choice element shall be a legal value within the constraints of the parameter field.

Attributes

- [choice.paramname](#) (optional) binds the element content to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Content

Text content containing one value that may legally be used for the specified parameter field(s).

Parents

- [parameter](#)

5.8 date

attribute

The date attribute shall contain the date the module was created. The date must be the same in any instance of the document.

Value: Shall conform to [xsd:date](#) (YYYY-MM-DD format based on [ISO-DATE])

Parents

- [device](#)
- [behaviorset](#)
- [languageset](#)

5.9 DDL

element

DDL is the root element of all DDL documents.

A DDL document shall fit the XML “document” production. The single root element shall be DDL.

DDL defines three module types: [device](#), [behaviorset](#) and [languageset](#) (see: DDL Modules – Devices, Behaviorsets and Languagesets). Each is defined in an element of the corresponding name. A DDL element shall contain exactly one module.

Attributes

- [version](#) (required)
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children

- Exactly *one* from [device](#) or [behaviorset](#) or [languageset](#)

Parents

None: [DDL](#) is always the root element.

5.10 device

element

The device element is a DDL module (see: DDL Modules – Devices, Behaviorsets and Languagesets) and follows all the requirements for modules.

Each device definition shall contain one or more root properties (with nested sub-properties as required) each of which forms the root of a property tree.

Attributes

- [UUID](#) (required) the unique identifier for this device type
- [provider](#) (required) the organization that published this description
- [date](#) (required) the publication date of this description
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children (in order)

1. [UUIDName](#) (zero or more)
2. [parameter](#) (zero or more)
3. [label](#) (zero or one)
4. [alternatefor](#) (zero or more)
5. [extends](#) (zero or more)

6. [useprotocol](#) (one or more)
7. Either [property](#) or [includedev](#) or [propertypointer](#) (one or more in any combination)

Parents

- [DDL](#)

5.11 extends

empty element

Identifies a module that is extended by the current one.

One module extends another when the extended module includes all the information of the original in unchanged form, but also includes further information. Rules for extension are separately defined for each module type.

Attributes

- [UUID](#) (required) the UUID the current module is extending.
- [UUID.paramname](#) (optional) binds the UUID attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Parents

- [device](#)
- [behaviorset](#)
- [languageset](#)

5.12 hd

element

The heading of a section.

Attributes

- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Content

Text content containing the heading.

Parents

- [section](#)

5.13 includedev

element

Includedev marks the point at which a sub device is included (see: Statically Included Device Descriptions).

Attributes

- [UUID](#) (required) the UUID of the device being included.
- [UUID.paramname](#) (optional) binds the UUID attribute to the named parameter.
- [array](#) (optional) if this is an array of sub devices.

- [array.paramname](#) (optional) binds the array attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children (in order)

1. [label](#) (zero or one)
2. [protocol](#) (zero or more) rules depend on specific protocols
3. [setparam](#) (zero or more)

Parents

- device
- property

5.14 key

attribute

The key identifying a string. Shall conform to the NCName production of the XML Namespace recommendation [XMLnames] (this is the same as the XML “Name” production but with colons disallowed).

Keys are used for lexical matching. They are case sensitive and shall be unique in the context in which they are used. See String Resources for language and key matching rules.

Parents

- [label](#) – key identifies the string to use as the label value
- [string](#) – key identifies this string within the specified language

5.15 label

element

Labels may be assigned to many elements in DDL. A label is intended for human consumption and should indicate the function of its parent element. Labels may take one of two forms. An immediate label contains the label text as its content. A referenced label contains both a string key and a languageset attribute that reference a string (or set of strings in different languages) that contains the text of the label.

A label shall have either immediate text content or both key and set attributes. If shall not have both content and attributes.

Attributes

- [key](#) (required if no text content, forbidden otherwise) the key identifying a string or strings (see: String Resources).
- [key.paramname](#) (optional) binds the key attribute to the named parameter.
- [label.paramname](#) (optional) binds the element content to the named parameter.
- [set](#) (required if no text content, forbidden otherwise) the UUID of the languageset in which the string(s) may be found.
- [set.paramname](#) (optional) binds the set attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Content

Shall fit [xsd:string](#) – (required if no attributes present, forbidden otherwise) the immediate text value of the

label.

Parents

- [behaviordef](#)
- [behaviorset](#)
- [device](#)
- [includedev](#)
- [language](#)
- [languageset](#)
- [parameter](#)
- [property](#)

5.16 lang

attribute

Identifies the language used in all [strings](#) in a [language](#) element.

The values of the attribute are language identifiers as defined by *Tags for the Identification of Languages* [LANG-TAG].

No two languages within a languageset shall have the same value for their lang attribute. (the value of lang must be unique within the [languageset](#)).

Value: Shall conform to [xsd:language](#)

Parents

- [language](#)

5.17 language

element

A language contains a set of [string](#) definitions in a particular language that is identified by its [lang](#) attribute. It may optionally specify an alternative language within the same languageset that should be searched if no string with a specified [key](#) is found in this language.

The [altlang](#) attribute points to another language in the same [languageset](#) by matching this language's [altlang](#) attribute with the [lang](#) attribute on another language. It is an error if there is no matching language element in the same languageset.

If no [altlang](#) attribute is present then this language must contain a [string](#) element for every [key](#) present in the languageset.

The use of altlang as a pointer can create chains of languages. It is an error if a language points to itself, whether directly or indirectly.

Attributes

- [lang](#) (required)
- [altlang](#) (optional)
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children (in order)

1. [label](#) (zero or one)
2. [string](#) (one or more)

Parents

- [languageset](#)

5.18 languageset

element

A languageset is a container for string resources with variants in multiple languages (see: String Resources). Multilingual string resources are available for labeling the description and also for reference from within devices.

String resources are grouped into [language](#) elements.

The languageset element groups one or more [language](#) elements each of which defines replacement text for the same set of keys but in a different language. Thus given a single key, the application may choose the replacement text according to the language preferences of the user. Each key potentially has replacement text in each language declared within the languageset.

Each languageset is a DDL module and bears a UUID that is used by elements that use its string resources to identify it. The set of keys defined by a languageset is the union of the keys defined by all the strings within it.

Attributes

- [UUID](#) (required) the unique identifier for this languageset
- [provider](#) (required) the organization that published this languageset
- [date](#) (required) the publication date of this languageset
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children (in order)

1. [UUIDname](#) (zero or more)
2. [label](#) (zero or one)
3. [alternatefor](#) (zero or more)
4. [extends](#) (zero or more)
5. [language](#) (one or more)

Parents

- [DDL](#)

5.19 maxinclusive

element

Specifies a maximum value restriction for a numeric parameter value. (see: Statically Included Device Descriptions and Parametric Devices).

If a maxinclusive element is present within a [parameter](#) element then no [choice](#) or [refinement](#) element shall be present in the same [parameter](#).

maxinclusive shall only be used for parameter fields that have a numeric value.

The value given by a maxinclusive element shall be a legal value within the constraints of the parameter fields to which the parameter applies and also shall conform to any restrictions imposed on the value of the fields by other elements either in the included device or in the same [parameter](#) element.

A maxinclusive value shall not be less than the value of any [mininclusive](#) present.

Attributes

- [maxinclusive.paramname](#) (optional) binds the element content to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Content: Text content containing a numeric value in the same format as the field being restricted.

Parents

- [parameter](#)

5.20 mininclusive

element

Specifies a minimum value restriction for a numeric parameter field. (see: Statically Included Device Descriptions and Parametric Devices).

If a mininclusive element is present within a [parameter](#) element then no [choice](#) or [refinement](#) element shall be present in the same [parameter](#).

mininclusive shall only be used for parameter fields that have a numeric value.

The value given by a mininclusive element shall be a legal value within the constraints of the parameter fields to which the parameter applies and also shall conform to any restrictions imposed on the value of the fields by other elements either in the included device or in the same [parameter](#) element.

A mininclusive value shall not be greater than the value of any [maxinclusive](#) present.

Attributes

- [mininclusive.paramname](#) (optional) binds the element content to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Content: Text content containing a numeric value in the same format as the field being restricted.

Parents

- [parameter](#)

5.21 name

attribute

A name that matches the NCName production of the XML Namespace recommendation [XMLnames] (this is the same as the XML “Name” production but with colons disallowed).

Names are used in various elements and special restrictions apply to some.

Names are used for lexical matching. They are case sensitive and shall be unique in the context in which they are used. e.g. a behavior name must be unique within a behaviorset, but may match the name of a behavior from another set or the name of a string that exists in a different context.

Value: Shall conform to [xsd:NCName](#)

Parents

- [behavior](#)
- [behaviordef](#)
- [parameter](#)
- [protocol](#)
- [refines](#)
- [setparam](#)
- [useprotocol](#)
- [UUIDname](#) – length of name in a UUIDname shall be 32 characters or less.

5.22 p

element

A paragraph of text (in a behavior definition). In presenting behaviors, processors may (and should) freely “fold” whitespace and word-wrap text in paragraphs to suit the presentation medium, unless the paragraph carries an [xml:space](#) attribute with the value “preserve”, in which case the paragraph should be presented as is with no formatting.

Attributes

- [xml:space](#) (optional)
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Content: Text content is the text of the paragraph.

Parents

- [section](#)

5.23 parameter

element

This element declares a parameter of a device for which a substitution value may be provided when the device is included as a template (see: Statically Included Device Descriptions and Parametric Devices). When the value for the parameter is provided on inclusion, it is applied to all the included fields that are bound to the parameter name using [paramname](#) attributes.

The parameter element declares the parameter near the start of a device and provides a name for it. It may optionally contain [choice](#), [refinement](#), [mininclusive](#) or [maxinclusive](#) elements restricting the range of values the parameter may take.

Attributes

- [name](#) (required) the name of the parameter
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children (in order):

1. [label](#) (zero or one)
2. Mutually exclusive options:

either [choice](#) (zero or more) or [refinement](#) (zero or more) or (in order):

1. [mininclusive](#) (zero or one)
2. [maxinclusive](#) (zero or one)

Parents

- [device](#)

5.24 .paramname

attribute

This is not a single attribute but is a suffix for an attribute used to bind a field to a parameter – see: Parametric Devices. The first part of the attribute name is the name of the attribute or element to be bound to the parameter. For example, attribute [array](#) can be parametrized by declaring a [array.paramname](#) attribute on the same element. If the field to be bound is the content of an element then the first part of the .paramname attribute shall be the same as the element name.

Value: Shall conform to [xsd](#):NCName

Parents

- [alternatefor](#) (UUID.paramname)
- [behavior](#) (name.paramname, set.paramname)
- [choice](#) (choice.paramname)
- [extends](#) (UUID.paramname)
- [includedev](#) (UUID.paramname, array.paramname)
- [label](#) (label.paramname, key.paramname, set.paramname)
- [maxinclusive](#) (maxinclusive.paramname)
- [mininclusive](#) (mininclusive.paramname)
- [property](#) (array.paramname, valuetype.paramname, shareddefine.paramname)
- [propertypointer](#) (ref.paramname)
- [protocol](#) (name.paramname)
- [setparam](#) (setparam.paramname)
- [refinement](#) (refinement.paramname)
- [useprotocol](#) (name.paramname)
- [value](#) (value.paramname, type.paramname)

Note

Additional parametrizable fields may be specified within protocol elements whose content is separately declared.

5.25 property

element

The property is the basic building block of device structure. Each property may itself have multiple properties nested within it. Every property has zero or one value.

Parents

- [device](#)

- [property](#)

Attributes

- [array](#) (optional) if not present defaults to 1 (a single value).
- [array.paramname](#) (optional) binds the array attribute to the named parameter.
- [valuetype](#) (required). Permitted values are:
 - NULL
 - network
 - implied
 - immediate
- [valuetype.paramname](#) (optional) binds the valuetype attribute to the named parameter. Note parametrizing valuetype is of limited use since changing it in most cases requires changes to the descendent XML structure that cannot be done using parameters.
- [shareddefine](#) (optional) if set “true” or “arraycommon” declares that this property is a defining declaration that may be referenced from elsewhere as a shared property. When set to “true” or “arraycommon” the property must also have a valid [xml:id](#) attribute.
- [shareddefine.paramname](#) binds the shareddefine attribute to the named parameter.
- [xml:id](#) (optional, required for defining declarations of shared properties) used to reference this element using ID/IDREF matching.

Content: depends on the valuetype attribute:

5.25.1 valuetype="NULL"

Children (in order):

1. [label](#) (optional)
2. [behavior](#) (one or more)
3. [protocol](#) (optional).
4. *Either* [property](#) *or* [includedev](#) *or* [propertypointer](#) (zero or more in any combination)

5.25.2 valuetype="network"

Children (in order):

1. [label](#) (optional)
2. [behavior](#) (one or more)
3. [protocol](#) (required).
4. *Either* [property](#) *or* [includedev](#) *or* [propertypointer](#) (zero or more in any combination)

5.25.3 valuetype="implied"

Children (in order):

1. [label](#) (optional)
2. [behavior](#) (one or more)
3. [protocol](#) (optional).
4. *Either* [property](#) *or* [includedev](#) *or* [propertypointer](#) (zero or more in any combination)

5.25.4 valuetype="immediate"

Children (in order):

1. [label](#) (optional)

2. [behavior](#) (one or more)
3. [protocol](#) (optional).
4. [value](#) (one or more)
5. *Either* [property](#) *or* [includedev](#) *or* [propertypointer](#) (zero or more in any combination)

5.26 propertypointer

empty element

A propertypointer is a shared property whose defining declaration is elsewhere. It is permitted in any place that a property is valid and represents the same property as the one to which it points. This allows the same property to be child of multiple parents that are in different places in the device. It is permitted for a propertypointer to reference another propertypointer provided that such a chain of references resolves to a full property declaration eventually.

The target of the ref attribute of a propertypointer shall be a property that has a shareddefine attribute whose value is either “true” or “arraycommon”.

Parents

- [device](#)
- [property](#)

Attributes

- [ref](#) (required). Identifies the defining declaration of this property.
- [ref.paramname](#) (optional) binds the ref attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

5.27 protocol

element

Protocol elements contain whatever information is required to enable processors to access networked property values. Their content is of necessity dependent on the [access protocol](#)(s) used.

Protocol elements may optionally be used within properties that do not have network values and within [includedev](#) to provide other protocol information as necessary – this is usually to specify rules applying to child properties. It is illegal for a protocol element to directly identify a networked value for a property unless that property has been declared with a value type of “network”.

Definition of the content of protocol is a large part of interfacing DDL to a particular [access protocol](#). Any protocol using DDL must specify how this is done.

For validation, a more complete schema definition for protocol tailored to an individual access protocol is often desirable. However, should definitions be written using multiple protocols, those definitions might prevent validation.

Attributes

- [name](#) (required) shall match an ESTA defined protocol name [ESTA-IDs]. Shall match a protocol identified by a useprotocol element in the same device.
- [name.paramname](#) (optional) binds the name attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children: any – depends on definitions supplied with specific protocols.

Parents

- [property](#)
- [includedev](#)

5.28 provider

attribute

The provider attribute shall unambiguously indicate the organization that created and maintains the definition. This is not necessarily the same as the organization that produces the equipment (for example a definition may be reused). The preferred form for the provider attribute is a URL that clearly identifies the organization and the appropriate department or section.

Value: Should conform to [xsd:anyURI](#)

Parents

- [device](#)
- [behaviorset](#)
- [languageset](#)

5.29 ref

attribute

Contains a reference that points to a property.

Value: Shall conform to the syntax and rules for extended references and references between arrays (see: Use of XML Identifiers to Reference Device Model Features).

Parents

- [propertypointer](#)

5.30 refinement

element

This establishes a restriction on the permissible values for a parameter (see: Statically Included Device Descriptions, Parametric Devices). It shall only be used when the parametrized field identified is a [behavior name](#). refinement specifies a base behavior. Any value provided for the parameter field (including any default behavior specified in-place) shall be a refinement of this base behavior.

If a refinement element is present within a [parameter](#) element then no [choice](#), shall be present in the same parameter. (behavior name attributes are not numeric so [mininclusive](#) and [maxinclusive](#) cannot apply).

The value given by a refinement element shall be a legal value within the constraints of the parameter field and also shall conform to any restrictions imposed on the value of the field by other elements either in the included device or in the same [parameter](#) element.

Attributes

- [refinement.paramname](#) (optional) binds the element content to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Content. Text content shall conform to [xsd:NCName](#) and shall match a behavior name within the

[behaviorset](#) identified by the target [behavior](#) element.

Parents

- [parameter](#)

5.31 refines

empty element

Every behavior may be a refinement of one or more existing definitions (see: Behavior Derivation and Refinement). [refines](#) identifies a single behavior definition that the current one derives from. The [refines](#) element is a reference to a [behaviordef](#) using exactly the same syntax as the [behavior](#) element. In order for the reference to be unambiguous, both the [behaviorset](#) and the name of the [behavior](#) within that set must be given.

Attributes

- [name](#) (required) the name of the behavior identified
- [set](#) (required) the UUID of the behaviorset within which the behavior definition occurs.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Parents

- [behaviordef](#)

5.32 section

element

Behavior definitions are textual descriptions that can be verbose. To allow some structuring of the description it is contained in sections that may be nested recursively. Each section has an optional heading and contains any number of freely intermixed sections and paragraphs.

Attributes

- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children (in order)

1. [hd](#) (optional)
2. one or more of: [section](#) and/or [p](#)

Parents

- [behaviordef](#)
- [section](#)

5.33 set

attribute

Identifies (by UUID) a behaviorset or languageset. The format and rules for the value are identical to [UUID](#).

Parents

- [refines](#) – set must identify a [behaviorset](#)
- [behavior](#) – set must identify a [behaviorset](#)

- [label](#) – set must identify a [languageset](#)

5.34 setparam

element

Specifies the value a parameter is to take, when a sub device is included in a description (see: Parametric Devices).

Attributes

- [name](#) (required) the name of the parameter – shall match the name of a [parameter](#) element within the [device](#) identified by the [includedev](#) containing this setparam element.
- [setparam.paramname](#) (optional) binds the element content to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Content. Text content is the value to be substituted for the named parameter in this instance of the template device.

Parents

- [includedev](#)

5.35 sharedefine

attribute

Used to identify the defining declaration of a property that is or may be shared(see: Shared Properties).

Value

Shall conform to [xsd:boolean](#). If *true*, then the property must also carry an [xml:id](#) attribute.

Legal values are:

false

The property cannot be referenced by a [propertypointer](#) as a defining declaration

true

This is a defining declaration of a shared property and may be the target of [propertypointer](#) declarations elsewhere in the device. The property must also carry an [xml:id](#) attribute.

arraycommon

This is a single property despite any array declarations on its parents or ancestors(see: Special Case – Property Shared Across an Array). Anywhere that this property is contained within an array, it is common to all items of that array. This may also be a defining declaration of a shared property and may be the target of [propertypointer](#) declarations elsewhere in the device. The property must also carry an [xml:id](#) attribute.

Default: If no sharedefine attribute is present then the default value false shall be used.

Parents

- [property](#)

5.36 string

element

A single text string resource. It is located by its key attribute.

The key attribute shall be unique within the enclosing [language](#).

Attributes

- [key](#) (required) the key identifying this string.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children: Text content is the text of string in the language of the parent [language](#) element.

Parents

- [language](#)

5.37 type

attribute

Specifies the type of a property [value](#) – signals to a processor how to parse the text representation.

Value

Legal values are:

uint

Value shall conform to xsd:unsignedInt. The value represents an integer within the range 0..4294967295.

sint

Value shall conform to xsd:int. The value represents an integer within the range -2147483648..+2147483647.

float

Value shall conform to xsd:double. The value represents an IEEE double precision floating point number.

string

The property value should be parsed or processed as a (Unicode) text string.

object

The property value is an arbitrary sized binary object that is not a regular integer. Immediate values shall be represented as a sequence of octets with each octet being represented as a pair of hexadecimal nibbles with the most significant nibble first. For readability, spaces, periods, commas or hyphens may be included between octets. Where a property is of this type, the behavior description must specify any byte ordering and formatting conventions necessary to interpret the value. It is recommended that network byte order be used unless there are overriding reasons to use other conventions.

Example 14: Immediate values of type “object”

```
<property
  type="object"
  value="1d2d5369 3f1809a6 07dcbb18 510fe564 8b65104d"
```

```

...
/>
<property
  type="object"
  value="b6.6b.b9.93.2f.44"
...
/>

```

Parents

- [value](#)

5.38 useprotocol

empty element

Identifies a protocol that may be used to access a device. It is illegal for a [protocol](#) element within a description to name a protocol that has not been declared in a [useprotocol](#) element within the same device.

Attributes

- [name](#) (required) shall match an ESTA defined protocol name [ESTA-IDs].
- [name.paramname](#) (optional) binds the name attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Parents

- [device](#)

5.39 UUID

attribute

A UUID attribute shall either contain a literal UUID value or a UUID name that has been assigned using the [UUIDname](#) element and that is in scope.

Literal values shall be written in the format given in [UUID]. All hexadecimal letters shall be in lower case but parsers should accept either lower case or upper case.

Note

Module identifiers are also referred to as [DCIDs](#) in this specification and elsewhere. The attribute name is always UUID.

Note

Applications may distinguish the format used by the length of the string. A literal value always has 36 characters while a name is required to be less than this.

Example 15: Examples of UUID values

```

UUID="3f399f5e-ad01-11d9-8525-000d613667e2"
UUID="xyz" <!-- named UUID UUIDname -->

```

Value: Shall fit [xsd:NMTOKEN](#) with maxLength = 36.

Parents

- [device](#) (must be literal UUID since UUIDname cannot be in scope).
- [behaviorset](#) (must be literal UUID since UUIDname cannot be in scope).
- [includedev](#) (may be explicit UUID or UUID name).
- [languageset](#) (must be literal UUID since UUIDname cannot be in scope).
- [UUIDname](#) (must be literal UUID – this element defines a corresponding UUID name).
- [alternatefor](#) (may be explicit UUID or UUID name).
- [extends](#) (may be explicit UUID or UUID name).

5.40 UUIDname

empty element

UUIDs are unwieldy for reasons of size and readability. A set of UUIDname elements at the start of a [module](#), allow short readable names to be assigned to UUIDs. Each UUIDname associates a single name with a UUID. The scope of a UUIDname is the entire [module](#) in which it occurs, with the exception of the attribute values of the root element of the module.

It is illegal for two UUIDname elements within the same module to have the same value for their name attribute.

Attributes

- [UUID](#) (required) is the UUID to be assigned the name. Shall be specified explicitly Named UUIDs are not permitted.
- [name](#) (required) shall conform to [xsd:NCname](#) with maxlength = 32.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Parents

- [device](#)
- [behaviorset](#)
- [languageset](#)

5.41 value

element

Contains the value of an immediate [property](#) (one whose value is declared within the description). If the [property](#) is within an array, it may contain multiple value elements – see Immediate Values for Array Properties.

Attributes

- [type](#) (required) the type of value contained
- [type.paramname](#) (optional) binds the type attribute to the named parameter.
- [value.paramname](#) (optional) binds the element content to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Content. Text content is the textual representation of the value of the parent property. See [type](#) for format.

Parents

- [property](#)

5.42 valuetype

attribute

Specifies the value type of a property (see: Four Locations for Property Value).

Value

Legal values are:

NULL

The property has no value – only sub properties

immediate

The value of the property is given within the description in a [value](#)

network

The property value is accessible over the network. One or more [protocol](#) elements contain the details of how to access it.

implied

The property has a hidden value that is not directly accessible. In most cases this value is driven by other properties (see: Meaning of Property Values – Driven Properties) and in some cases the value may be inferred. For bidirectional protocols, use of implied properties is discouraged – it is generally better to make it readable. For unidirectional protocols such as DMX512 [DMX512] implied values are often inevitable.

Parents

- [property](#)

5.43 version

attribute

This is the DDL version. The version number “1.1” shall be used to indicate conformance to this version of this specification; it is an error for a document to use the value “1.1” if it does not conform to this version of this specification. This construct is provided, as a means to allow the possibility of automatic version recognition, should it become necessary. Processors should report an error if they receive documents labeled with versions they do not support.

Value shall be “1.1”

Parents

- DDL

5.44 xml:space

attribute

Follows the semantics and rules defined in the XML specification [XML]. See [p](#) element for details.

Value

Legal values are:

default

default behavior is to “fold” whitespace and wrap words

preserve

text is preformatted and should be presented “as is”

Parents

- [p](#)

5.45 xml:id

attribute

Follows the semantics and rules defined in [xml-id] specification. See Use of XML Identifiers to Reference Device Model Features.

Value. The value shall match the NCName definition in the XML namespace specification [XMLnames].

Parents. xml:id may be used on any element within DDL.

6 Interface to the Access Protocol

Protocols vary wildly in their notations, and capabilities. In order for DDL to describe real devices using any given protocol, an interface needs to be defined between DDL and that protocol. This section gives guidelines for what needs to be defined in order to apply DDL to a particular protocol. A concrete example is also given in [DMP] that defines the interface to a real protocol.

All protocol specific information must be specified within [protocol](#) elements that identify the protocol they refer to. The [protocol](#) element must be customized for each protocol to be interfaced.

6.1 Access to Network Properties

6.1.1 Network Property Values

Property values may either be constant immediate values embedded in the description or may be accessed via the network or data link. In the latter case a protocol element shall be provided that must provide sufficient information for a controller using that protocol to access the value. This includes not just network location of the property or the message type(s) needed to access it, but frequently such things as “on the wire” data size and representation and read/write characteristics.

6.1.2 Network Data Type and Size

In many protocols data representation and encoding is frequently best expressed as a behavior, however, a minimal processor should be able to read or write and pass on property values without knowledge of specific behaviors. Therefore, the networked value reference elements must specify any items such as the size, read/write accessibility and so on that are required to access and handle the data without necessarily interpreting it.

6.2 Necessary Definitions and Restrictions

All items in this section must be customized or defined for a protocol interface.

6.2.1 Protocol Declaration

A protocol to be used with DDL needs to have a key-name registered with ESTA.

Format of protocol identifiers and processes for registration are given in [ESTA-IDs].

6.2.2 Network Property Access Mappings

The protocol interface must define how the protocol is used with the information given in a declaration to examine and/or modify the values of a network property.

6.3 Behaviors

While the behavior mechanism is part of DDL, specific behaviors are not. A base set of behaviors are provided for [DMP] that may be reused or adapted as necessary with other protocols.

Appendix A. Generic DTD DDL

This DTD (refer to [XML]) is provided for reference purposes. See DTD, Schemas and Validation for discussion of schemas and validation. Note that the [protocol](#) element is declared with content ANY, but that any elements for specific protocols contained within it will nevertheless need to be declared before this DTD can be used.

The public identifier for this DTD shall be:

```

“-//ESTA//DTD Device Description Language 1.1//EN”
<!--
  DTD for Device Description Language
  no specific protocol declared
-->
<!ELEMENT DDL ((behaviorset | device | languageset))>
<!ATTLIST DDL
  version ( 1.1 ) #REQUIRED
  xml:id ID #IMPLIED
>
<!-- common module content -->
<!ELEMENT label (#PCDATA)>
<!ATTLIST label
  label.paramname NMTOKEN #IMPLIED
  set NMTOKEN #IMPLIED
  set.paramname NMTOKEN #IMPLIED
  key NMTOKEN #IMPLIED
  key.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
>
<!ELEMENT alternatefor EMPTY>
<!ATTLIST alternatefor
  UUID NMTOKEN #REQUIRED
  UUID.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
>
<!ELEMENT extends EMPTY>
<!ATTLIST extends
  UUID NMTOKEN #REQUIRED
  UUID.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
>
<!ELEMENT UUIDname EMPTY>
<!ATTLIST UUIDname
  name NMTOKEN #REQUIRED
  UUID NMTOKEN #REQUIRED
  xml:id ID #IMPLIED
>
<!-- languageset module -->
<!ELEMENT languageset (
  UUIDname*, label?,
  alternatefor*,

```

```

    extends*,
    language+
)>
<!-- languageset content -->
<!-- ATTTLIST languageset
    UUID NMToken #REQUIRED
    provider CDATA #REQUIRED
    date NMToken #REQUIRED
    xml:id ID #IMPLIED
-->
<!-- language (label?, string*) -->
<!-- ATTTLIST language
    lang CDATA #REQUIRED
    altlang CDATA #IMPLIED
    xml:id ID #IMPLIED
-->
<!-- string (#PCDATA) -->
<!-- ATTTLIST string
    key NMToken #REQUIRED
    xml:id ID #IMPLIED
-->
<!-- behaviorset module -->
<!-- ELEMENT behaviorset (
    UUIDName*, label?,
    alternatefor*,
    extends*,
    behaviordef+
-->
<!-- ATTTLIST behaviorset
    UUID NMToken #REQUIRED
    provider CDATA #REQUIRED
    date NMToken #REQUIRED
    xml:id ID #IMPLIED
-->
<!-- behaviorset content -->
<!-- ELEMENT behaviordef (label?, refines*, section+) -->
<!-- ATTTLIST behaviordef
    name NMToken #REQUIRED
    xml:id ID #IMPLIED
-->
<!-- refines EMPTY -->
<!-- ATTTLIST refines
    set NMToken #REQUIRED
    name NMToken #REQUIRED
    xml:id ID #IMPLIED
-->
<!-- section (hd?, (section | p)+) -->
<!-- ATTTLIST section
    xml:id ID #IMPLIED
-->
<!-- hd (#PCDATA) -->
<!-- ATTTLIST hd

```

```

    xml:id ID #IMPLIED
>
<!ELEMENT p (#PCDATA)>
<!ATTLIST p
    xml:space (default | preserve) 'default'
    xml:id ID #IMPLIED
>
<!-- device module -->
<!ELEMENT device (
    UUIDname*, parameter*, label?,
    alternatename*,
    extends*,
    useprotocol+,
    (property | propertypointer | includedev)+
)>
<!ATTLIST device
    UUID NMTOKEN #REQUIRED
    provider CDATA #REQUIRED
    date NMTOKEN #REQUIRED
    xml:id ID #IMPLIED
>
<!-- parameter declarations -->
<!ELEMENT parameter (
    label?,
    ( choice*
      | refinement*
      | ( mininclusive?, maxinclusive?)
    )
)>
<!ATTLIST parameter
    name NMTOKEN #REQUIRED
    xml:id ID #IMPLIED
>
<!--parameter restrictions -->
<!ELEMENT choice (#PCDATA)>
<!ATTLIST choice
    choice.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
>
<!ELEMENT mininclusive (#PCDATA)>
<!ATTLIST mininclusive
    mininclusive.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
>
<!ELEMENT maxinclusive (#PCDATA)>
<!ATTLIST maxinclusive
    maxinclusive.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
>
<!ELEMENT refinement (#PCDATA)>
<!ATTLIST refinement
    refinement.paramname NMTOKEN #IMPLIED

```

ANSI E1.17-2010, ACN – Device Description Language (DDL)

```

    xml:id ID #IMPLIED
>
<!-- see below for useprotocol and other protocol dependent content -->
<!-- properties -->
<!ELEMENT property (
    label?,
    behavior+,
    value*,
    protocol*,
    (property | propertypointer | includedev)*
)>
<!ATTLIST property
    array CDATA #IMPLIED
    array.paramname NMTOKEN #IMPLIED
    valuetype (NULL | immediate | implied | network | common) #IMPLIED
    valuetype.paramname NMTOKEN #IMPLIED
    shareddefine (false | true | arraycommon) "false"
    shareddefine.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
>
<!ELEMENT behavior EMPTY>
<!ATTLIST behavior
    set NMTOKEN #REQUIRED
    set.paramname NMTOKEN #IMPLIED
    name NMTOKEN #REQUIRED
    name.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
>
<!ELEMENT value (#PCDATA)>
<!ATTLIST value
    value.paramname NMTOKEN #IMPLIED
    type (uint | sint | float | string | object) #REQUIRED
    type.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
>
<!ELEMENT propertypointer EMPTY>
<!ATTLIST propertypointer
    ref CDATA #REQUIRED
    ref.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
>
<!-- included devices and parameter instantiation -->
<!ELEMENT includedev (label?, protocol*, setparam*)>
<!ATTLIST includedev
    UUID NMTOKEN #REQUIRED
    UUID.paramname NMTOKEN #IMPLIED
    array CDATA #IMPLIED
    array.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
>
<!ELEMENT setparam (#PCDATA)>
<!ATTLIST setparam

```

ANSI E1.17-2010, ACN – Device Description Language (DDL)

```
name NMTOKEN #REQUIRED
setparam.paramname NMTOKEN #IMPLIED
xml:id ID #IMPLIED
>
<!-- Protocol dependent section -->
<!--
  The following elements have generic content models here
  They may be specialized as described for specific protocols
-->
<!ELEMENT useprotocol EMPTY>
<!ATTLIST useprotocol
  name CDATA #REQUIRED
  name.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
>
<!--
  Note:
  The name attribute shall be an ESTA registered name.
-->
<!ELEMENT protocol ANY >
<!ATTLIST protocol
  name CDATA #REQUIRED
  name.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
>
<!--
  Notes:
  The name attribute shall match the ESTA registered name
  of a protocol declared in a preceding useprotocol element.
  Content must conform to rules specified for the protocol
  identified by the name attribute.
  No further attributes permitted.
-->
<!-- End of protocol dependent section -->
```

Appendix B. DDL Interface to DMP

Device Management Protocol (DMP) and Device Description Language (DDL) are part of the ESTA ACN suite of protocols for control of entertainment technology equipment. They are fully described in the DMP specification [DMP] and the DDL Specification [DDL].

B.1 Relationship Between DDL Devices and DMP Components

Each DMP [component](#) (see [Arch]) that exposes properties is defined in DMP to be a device – in DMP there can only be one set of properties within a component and the factoring of these properties into multiple DDL devices is a function of DDL that is not visible in DMP. In DDL terms this is a single [appliance](#). The UUID [UUID] of its single root device is expected to be made available through whatever ACN Discovery method is applicable. The appliance may be described by any number of additional child devices and descendant devices. Thus given the root device class, the structure of the entire component can be examined by finding the child devices of the root device and then their children and so on.

B.2 Message Mappings

The values of network property elements in DDL correspond directly to DMP values that are accessed using the various forms of Get_Property, Set_Property and Event messages of DMP.

B.3 DMP Property Access

The characteristics necessary to access DMP properties are all specified within [protocol](#) elements using [propref_DMP](#) and [childrule_DMP](#). [propref_DMP](#) specifies the location and access characteristics of a DMP property while [childrule_DMP](#) does not specify any DMP property itself but sets the location origin which is used by the relative address syntax for sub properties and included devices.

B.3.1 Addressing

DMP properties are single values that have a one-to-one correspondence with DDL Network Accessed properties (see: Four Locations for Property Value). A DMP property is specified using a single scalar address range (0..4294967295).

A DMP address may be specified in DDL in a relative or absolute form.

B.3.1.4 Relative Addressing and Location Origin

To provide the address of a DMP property, a relative addressing scheme is available in DDL and is the preferred address syntax. DMP addresses or network properties are expressed as signed offsets relative to a location origin which is defined for each DDL property. For details see [loc](#) and [childrule_DMP](#).

The use of relative addressing means that any property tree or subtree in a device defines a pattern of locations for its properties that are relative to the location of the root of the subtree. This allows a subtree to be repeated elsewhere in the address space with a different location origin but no change at all to its description.

Note

The expression of DMP addresses in relative form in DDL does not affect the way the addresses must be specified in DMP messages and is not connected with any specific addressing modes provided by DMP. Any address specified in DDL must be resolved to an absolute address before it can be used in generating a DMP message.

B.3.1.5 Absolute Addressing

It is possible to specify DMP addresses (and location origins) in absolute form using [abs](#) but this is not recommended in normal circumstances because it restricts re-use of devices by inclusion or reference.

B.3.1.6 Non-network Properties

Properties that do not have a network value shall not contain a DMP address specification. However they may contain [childrule_DMP](#) to specify address characteristics for their children.

B.3.2 Property Arrays

In accordance with Arrays of Properties any property may be declared as an array. The DMP addresses of array properties shall be specified as follows:

For any property that has an array attribute with a value greater than one (any array property):

- If the array property has a DMP network value. The [propref_DMP](#) element specifying the DMP address shall have an [inc](#) attribute specifying how the DMP property address increments. This increment applies to iteration over the array specified on this property only and not to iteration over any array inherited from ancestor properties.
- If the array property has one or more descendant properties that have DMP network values, or if the array property includes devices (either statically included or attached by reference) that have or could have DMP network accessible properties; it shall contain a [childrule_DMP](#) element (within a protocol element) that includes an [inc](#) attribute specifying how the addresses of all child DMP properties increment. The increment specified by this [childrule_DMP](#) applies to iteration over the array count specified by the immediate property only (by the property element that is the grandparent of the [childrule_DMP](#) element). It does not apply to any other index of a multidimensional array.

Any property within an array for which `shareddefine="arraycommon"` is declared, is a single property (see: Special Case – Property Shared Across an Array). Its address is determined solely by the relevant [loc](#) and [abs](#) attributes of its [propref_DMP](#) element and the [childrule_DMP](#) elements of its ancestors without reference to their [inc](#) attributes. This is identical to the address its first element would have if it were left as an array.

B.3.3 Accessibility

DMP properties may be accessed using `Get_Property` and `Set_Property` and `Event`. However, it is not required that a particular property be accessible using all three messages. Attributes [read](#) and [write](#) and [event](#) indicate whether a property may be accessed using `Get_Property` and/or `Set_Property` and/or `Event` respectively. Note that event publishing may require auxiliary properties to be manipulated (e.g. to set publishing frequency) and these are described in behaviors.

B.3.4 Property size and variable size properties

DMP provides two transfer methods for property values: fixed length and variable length.

Variable length transfers encode the value length within the message and can vary in length from one message to another depending on the value being transferred. This carries an overhead in order to encode the length. Variable length transfers are suitable for text strings and other items whose intrinsic size varies.

Fixed length transfers do not encode the length of the value so this must be known. Fixed length transfers are suitable for items such as integers, floating point quantities, or objects that have a fixed format.

The value of any particular DMP property must always be transferred using the same method and if fixed length, using the same number of octets irrespective of the value, message or any other factor.

The DDL description of a property shall specify which transfer method must be used for that property and for fixed length transfers, the size (the number of octets used). These are specified using the [size](#) and [varsize](#) attributes.

B.3.5 Additional Characteristics of DMP Properties

With just the information in [DMP protocol](#) elements, an application can store and retransmit properties but cannot generate or modify them in any way since they may use different internal representations (e.g. floating point vs integer vs unordered bitmap).

Any higher interpretation placed on a property value must be discovered from its behavior – behaviors for common data types are defined in the core ACN behavior set.

B.3.6 Byte Ordering

While specific encodings are specified and described in DDL behaviors, it is forbidden to create behaviors that specify transmission of property values in non network byte order except in the exceptional case where an existing encoding is used for a specific function and where that encoding is standardized by a recognized standards body for use across platforms of both byte orders.

B.4 DMP Element and Attribute Reference

B.4.1 useprotocol (DMP devices)

empty element

Identifies a protocol that may be used to access a device. For any description of DMP access to a device there shall be a useprotocol element with `name="ESTA.DMP"`. There may be additional useprotocol elements if the description also applies for other access protocols.

Attributes

- [name](#) (required) shall have the value “ESTA.DMP”.
- [name.paramname](#) (optional) binds the name attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Parents

- [device](#)

B.4.2 protocol (DMP devices)

element

In DMP a protocol element shall have the following content.

Attributes

- [name](#) (required) shall have the value “ESTA.DMP”
- [name.paramname](#) (optional) binds the name attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Children

- [propref_DMP](#) (required if parent is property with `valuetype="network"`, forbidden otherwise) specifies a network property.

- [childrule_DMP](#) (optional) specifies location origin of children of this property or of the root properties of an included device.

Parents

- [property](#)
- [includedev](#)

B.4.3 propref_DMP

empty element

Specifies the location and characteristics of a DMP property or property array.

Attributes

- [loc](#) (required)
- [loc.paramname](#) (optional) binds the loc attribute to the named parameter.
- [size](#) (required)
- [size.paramname](#) (optional) binds the size attribute to the named parameter.
- [abs](#) (optional)
- [abs.paramname](#) (optional) binds the abs attribute to the named parameter.
- [read](#) (optional)
- [read.paramname](#) (optional) binds the read attribute to the named parameter.
- [write](#) (optional)
- [write.paramname](#) (optional) binds the write attribute to the named parameter.
- [event](#) (optional)
- [event.paramname](#) (optional) binds the event attribute to the named parameter.
- [varsize](#) (optional)
- [varsize.paramname](#) (optional) binds the varsize attribute to the named parameter.
- [inc](#) (optional)
- [inc.paramname](#) (optional) binds the inc attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Parents

- [protocol](#)

B.4.4 childrule_DMP

empty element

Specifies the location origin for all child properties of the current property or included device ([includedev](#)). The location origin is used to specify DMP addresses using a relative form – see [loc](#).

If a property does not contain a childrule_DMP specification (inside a [protocol](#) element), then its children (both [property](#) and [includedev](#)) shall inherit its own location origin unchanged.

The location origin for the root properties in a device that is attached by reference (see: Sub Devices Attached by Reference) shall be received from the device-reference property that anchors the sub-device within the parent device as though they were direct children of that property.

The location origin for the root properties in a device that is statically included (see: Statically Included Device Descriptions) may be specified explicitly using [childrule_DMP](#) (inside [protocol](#)) in the [includedev](#).

element. If not specified explicitly the location origin shall be the same as would apply for a property at the point of inclusion.

The location origin for the root properties of the root device in an [appliance](#) shall be 0.

Attributes

- [loc](#) (optional)
- [loc.paramname](#) (optional) binds the loc attribute to the named parameter.
- [abs](#) (optional if loc is present, forbidden otherwise)
- [abs.paramname](#) (optional) binds the abs attribute to the named parameter.
- [inc](#) (optional)
- [inc.paramname](#) (optional) binds the inc attribute to the named parameter.
- [xml:id](#) (optional) used to reference this element using ID/IDREF matching.

Parents

- [protocol](#)

B.4.5 loc

attribute

Specifies a DMP location (address). DMP addresses are in the range 0..4294967295. This address may be either a property address ([propref_DMP](#)) or the location origin for child properties ([childrule_DMP](#)).

If [abs](#) is present on the same element and is true then this is an absolute DMP address and must fall within the legal range.

If [abs](#) is not present or is false then the DMP address is given in a relative form and is a signed offset of the DMP address relative to the location origin of the property or device (see: Relative Addressing and Location Origin).

Value

if [abs](#) is true

[loc](#) shall conform to [xsd:unsignedInt](#)

else

[loc](#) shall conform to [xsd:int](#)

Parents

- [propref_DMP](#)
- [childrule_DMP](#)

B.4.6 abs

attribute

If true abs specifies that the DMP location (address) given by [loc](#) is in an absolute form. If false or absent, then [loc](#) is in its relative form.

Value. Shall conform to xsd:boolean.

Default. If no abs attribute is present then the default value false shall be used.

Parents

- [propref_DMP](#)
- [childrule_DMP](#)

B.4.7 inc

attribute

When the property is a part of an array, inc specifies the increment in DMP address between each element.

Value. Shall conform to [xsd:unsignedShort](#)

Default. No default is provided. [inc](#) shall be explicitly included wherever an increment must be specified according to the rules in Property Arrays.

Parents

- [propref_DMP](#)
- [childrule_DMP](#)

B.4.8 read

attribute

If true the DMP property is readable (using a Get_Property message). If false or not present, the property may not be read.

Value. Shall conform to [xsd:boolean](#).

Default. If no read attribute is present then the default value false shall be used.

Parents

- [propref_DMP](#)

B.4.9 write

attribute

If true the DMP property is writable (using a Set_Property message). If false or not present, the property may not be written.

Value. Shall conform to [xsd:boolean](#).

Default. If no write attribute is present then the default value false shall be used.

Parents

- [propref_DMP](#)

B.4.10 event

attribute

If true the DMP property is capable of generating events (set up using a Subscribe_Event message). If false or not present, the property cannot generate events.

Note

A number of behaviors are defined in the ACN base behaviorset for properties that modify or control the generation of events and further behaviors may be defined. Simply sending a subscribe message may not be sufficient to actually receive events from a property.

Value. Shall conform to [xsd](#):boolean.

Default. If no event attribute is present then the default value false shall be used.

Parents

- [propref_DMP](#)

B.4.11 varsize**attribute**

If true the DMP property value is transferred using DMP's variable length encoding. If false or not present, the value is always transferred in the fixed number of octets given by [size](#) (see: Property size and variable size properties).

Value. Shall conform to xsd:boolean.

Default. If no varsize attribute is present then the default value false shall be used.

Parents

- [propref_DMP](#)

B.4.12 size**attribute**

Specifies the number of octets DMP uses to transfer the property value. If [varsize](#) is true then this is the maximum size of the data itself excluding the size indicator (see: Property size and variable size properties).

Value. Shall conform to [xsd](#):unsignedShort

Parents

- [propref_DMP](#)

B.5 Examples

These assume that an (non-existent) example behaviorset “f3347bac-3e20-437d-a69e-019f37a71288” has been assigned the name “foo.bset” in the parent device, and that the ACN base behaviorset “71576eac-e94a-11dc-b664-0017316c497d” has been assigned the name “acnbase.bset” (using [UUIDname](#)):

```
<device ...>
  <UUIDname UUID="f3347bac-3e20-437d-a69e-019f37a71288" name="foo.bset"/>
  <UUIDname UUID="71576eac-e94a-11dc-b664-0017316c497d"
name="acncore.bset"/>
  ...
</device>
```

B.5.1 Simple Property

Define a property that contains a 16 bit signed integer, that can be read and written at DMP address “10”.

Example B.5.1: DDL for a Simple Property

```
<property valuetype="network">
  <behavior set="foo" name="bar"/>
  <protocol name="ESTA.DMP">
    <propref_DMP
      loc="10"
      abs="true"
      size="2"
      read="true"
      write="true"
    />
  </protocol>
</property>
```

B.5.2 Array of Simple Properties

Define an array of 512 properties that are 8 bit unsigned integers, that can be written but not read, and live at 0—511 within the DMP address space:

Example B.5.2: DDL for Array of Simple Properties

```
<property valuetype="network" array="512">
  <behavior set="foo" name="bar"/>
  <protocol name="ESTA.DMP">
    <propref_DMP
      loc="0"
      abs="true"
      size="1"
      write="true"
      inc="1"
    />
  </protocol>
</property>
```

B.5.3 Included Device

Define an included sub device that has [DCID](#) “38763d0c-b0f9-11d9-8615-000d613667e2” and whose root properties have a location origin offset by 100 relative to the current location origin. (Current location origin + 100 is the location origin for the root properties within the sub-device.)

Example B.5.3: DDL for a Direct Device Reference

```
<includedev UUID="38763d0c-b0f9-11d9-8615-000d613667e2">
  <protocol name="ESTA.DMP">
    <childrule_DMP loc="100"/>
  </protocol>
</includedev>
```

B.5.4 Subdevice Attached by Reference

Define an attached sub device that has [DCID](#) “38763d0c-b0f9-11d9-8615-000d613667e2” and whose root properties have a location origin offset by 100 relative to the current location origin. (Current location origin

+ 100 is the location origin for the root properties within the sub-device.)

This example achieves the same logical structure as the previous example and is included for informative purposes and as an introduction to attachment. In general where the attached device is statically defined (using an immediate value) attachment by reference is less flexible than static inclusion.

Example B.5.4: DDL for a Direct Device Reference

```
<property valuetype="immediate">
  <behavior set="acnbase.bset" name="deviceRef"/>
  <value type="object">38763d0c-b0f9-11d9-8615-000d613667e2</value>
  <protocol name="ESTA.DMP">
    <childrule_DMP loc="100" />
  </protocol>
</property>
```

B.5.5 External Device References

Define a reference to a sub device that is not predefined. The [DCID](#) of the sub device is at address 20 while the location origin of the sub devices properties is DMP address 1000. Because address 20 allows events, the controller can be notified should the type of the sub-device be changed dynamically.

Example B.5.5: DDL for an External Device Reference

```
<property valuetype="network">
  <behavior name="deviceRef" set="acnbase.bset"/>
  <protocol name="ESTA.DMP">
    <propref_DMP
      loc="20"
      size="16"
      read="true"
      event="true"
    />
    <childrule_DMP
      loc="1000"
      abs="true"
    />
  </protocol>
</property>
```

B.5.6 Array of Sub Devices

Define a reference to an array of 96 identical sub devices that have the [DCID](#) “38763d0c-b0f9-11d9-8615-000d613667e2” with the first sub device's properties addresses starting at 500 and subsequent sub devices having properties starting at 1500, 2500, 3500 etc. through the DMP address space.

The “foo” parameter for all subdevices is set to “99”.

Example B.5.6: DDL for an Array of Direct Device References

```
<property valuetype="NULL" array="96">
  <behavior name="group" set="acnbase.bset"/>
  <includedev
    UUID="38763d0c-b0f9-11d9-8615-000d613667e2"
    array="96"
  >
  <protocol name="ESTA.DMP">
```

```

    <childrule_DMP
      loc="500"
      abs="true"
      inc="1000"
    />
  </protocol>
  <setparam name="foo">99</setparam>
</includedev>
</property>

```

B.5.7 Array of External Device References

Define a reference to an array of 16 sub devices that are not predefined. The DCIDs of the sub devices live in an array from addresses 11-26 while the sub devices themselves have properties starting at addresses 1024, 2048, 3072... etc.

Example B.5.7: DDL for an Array of External Device References

```

<property valuetype="network" array="16">
  <behavior name="deviceRef" set="acnbase.bset"/>
  <protocol name="ESTA.DMP">
    <propref_DMP
      loc="11"
      abs="true"
      inc="1"
      size="16"
      read="true"
      event="true"
    />
    <childrule_DMP
      loc="1024"
      inc="1024"
      abs="true"
    />
  </protocol>
</property>

```

Table B.1: Address space diagram

Address	Function
0 – 10	Available for other properties
11 – 26	Array of 16 DCIDs of sub devices
27 – 1023	Available for other properties
1024 – 2047	Address space for first sub device
2048 – 3071	Address space for second sub device
...	...
16384 – 17407	Address space for 16 th sub device
17408 – 4294967295	Available for other properties or sub devices

B.5.8 Array Example

This very simplified example using fictitious behaviors (similar to some DMP behaviors) declares an array of 96 dimmers (intensity behavior). Each intensity is a driven value that is produced by an array of 10 Highest Takes Precedence inputs (HTPdriver behavior). Each HTP input also has a “preheat” level in a (minLimit behavior), however the preheat level is shared across all the HTP inputs for each dimmer and therefore there are only 96 rather than 960 preheat properties.

Relative to the location origin specified for the top level property, the 96 intensity properties are at addresses 0..95. The 960 HTP inputs are at 200..209, 220..229, 240..249, etc. The 96 preheat levels are at locations 219, 239, 259 etc.

Example B.5.8: DDL for an Array of Direct Device References

```
<property array="96" valuetype="network">
  <label>Dimmer Array</label>
  <behavior name="intensity" set="foo.bset" />
  <behavior name="HTPdriver" set="foo.bset" />
  <protocol name="ESTA.DMP">
    <propref_DMP event="true" inc="1" loc="0"
      read="true" size="2" />
    <childrule_DMP inc="20" loc="200" />
  </protocol>
  <property array="10" valuetype="network">
    <label>HTP Array</label>
    <behavior name="HTPdriver" set="foo.bset" />
    <protocol name="ESTA.DMP">
      <propref_DMP inc="1" loc="0"
        read="true" size="2" write="true" />
    </protocol>
    <property valuetype="network" xml:id="preheat"
sharedefine="arraycommon">
      <label>Preheat</label>
      <behavior name="minLimit" set="foo.bset" />
      <protocol name="ESTA.DMP">
        <propref_DMP loc="19"
          read="true" size="2" write="true" />
      </protocol>
    </property>
  </property>
</property>
```

Appendix C. DTD for DMP Only

A much more complete DTD for use with the DMP protocol only is produced from the generic DDL version by adding declarations for the propref_DMP and childrule_DMP elements and tightening a few other declarations to restrict to one specific protocol.

The public identifier for this DTD shall be:

```
"-//ESTA//DTD Device Description Language 1.1//EN//DMP"
```

```
<!--
```

```
  DTD for Device Description Language using ESTA DMP protocol
```

ANSI E1.17-2010, ACN – Device Description Language (DDL)

```

Registered protocol name is "ESTA.DMP"
-->
<!ELEMENT DDL ((behaviorset | device | languageset))>
<!ATTLIST DDL
  version ( 1.1 ) #REQUIRED
  xml:id ID #IMPLIED
>
<!-- common module content -->
<!ELEMENT label (#PCDATA)>
<!ATTLIST label
  label.paramname NMTOKEN #IMPLIED
  set NMTOKEN #IMPLIED
  set.paramname NMTOKEN #IMPLIED
  key NMTOKEN #IMPLIED
  key.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
>
<!ELEMENT alternatefor EMPTY>
<!ATTLIST alternatefor
  UUID NMTOKEN #REQUIRED
  UUID.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
>
<!ELEMENT extends EMPTY>
<!ATTLIST extends
  UUID NMTOKEN #REQUIRED
  UUID.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
>
<!ELEMENT UUIDname EMPTY>
<!ATTLIST UUIDname
  name NMTOKEN #REQUIRED
  UUID NMTOKEN #REQUIRED
  xml:id ID #IMPLIED
>
<!-- languageset module -->
<!ELEMENT languageset (
  UUIDname*, label?,
  alternatefor*,
  extends*,
  language+
)>
<!ATTLIST languageset
  UUID NMTOKEN #REQUIRED
  provider CDATA #REQUIRED
  date NMTOKEN #REQUIRED
  xml:id ID #IMPLIED
>
<!-- languageset content -->
<!ELEMENT language (label?, string*)>
<!ATTLIST language

```

```

    lang CDATA #REQUIRED
    altlang CDATA #IMPLIED
    xml:id ID #IMPLIED
>
<!ELEMENT string (#PCDATA)>
<!ATTLIST string
    key NMTOKEN #REQUIRED
    xml:id ID #IMPLIED
>
<!-- behaviorset module -->
<!ELEMENT behaviorset (
    UUIDname*, label?,
    alternatefor*,
    extends*,
    behaviordef+
)>
<!ATTLIST behaviorset
    UUID NMTOKEN #REQUIRED
    provider CDATA #REQUIRED
    date NMTOKEN #REQUIRED
    xml:id ID #IMPLIED
>
<!-- behaviorset content -->
<!ELEMENT behaviordef (label?, refines*, section+)>
<!ATTLIST behaviordef
    name NMTOKEN #REQUIRED
    xml:id ID #IMPLIED
>
<!ELEMENT refines EMPTY>
<!ATTLIST refines
    set NMTOKEN #REQUIRED
    name NMTOKEN #REQUIRED
    xml:id ID #IMPLIED
>
<!ELEMENT section (hd?, (section | p)+)>
<!ATTLIST section
    xml:id ID #IMPLIED
>
<!ELEMENT hd (#PCDATA)>
<!ATTLIST hd
    xml:id ID #IMPLIED
>
<!ELEMENT p (#PCDATA)>
<!ATTLIST p
    xml:space (default | preserve) 'default'
    xml:id ID #IMPLIED
>
<!-- device module -->
<!ELEMENT device (
    UUIDname*, parameter*, label?,
    alternatefor*,
    extends*,

```

```

    useprotocol+,
    (property | propertypointer | includedev)+
  )>
<!-- ATTLIST device
  UUID NMTOKEN #REQUIRED
  provider CDATA #REQUIRED
  date NMTOKEN #REQUIRED
  xml:id ID #IMPLIED
-->
<!-- parameter declarations -->
<!-- ELEMENT parameter (
  label?,
  ( choice*
    | refinement*
    | ( mininclusive?, maxinclusive?)
  )
-->
<!-- ATTLIST parameter
  name NMTOKEN #REQUIRED
  xml:id ID #IMPLIED
-->
<!-- parameter restrictions -->
<!-- ELEMENT choice (#PCDATA)>
<!-- ATTLIST choice
  choice.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
-->
<!-- ELEMENT mininclusive (#PCDATA)>
<!-- ATTLIST mininclusive
  mininclusive.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
-->
<!-- ELEMENT maxinclusive (#PCDATA)>
<!-- ATTLIST maxinclusive
  maxinclusive.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
-->
<!-- ELEMENT refinement (#PCDATA)>
<!-- ATTLIST refinement
  refinement.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
-->
<!-- see below for useprotocol and other protocol dependent content -->
<!-- properties -->
<!-- ELEMENT property (
  label?,
  behavior+,
  value*,
  protocol*,
  (property | propertypointer | includedev)*
-->
<!-- ATTLIST property

```

ANSI E1.17-2010, ACN – Device Description Language (DDL)

```

array CDATA #IMPLIED
array.paramname NMTOKEN #IMPLIED
valuetype (NULL | immediate | implied | network | common) #IMPLIED
valuetype.paramname NMTOKEN #IMPLIED
shareddefine (false | true | arraycommon) "false"
shareddefine.paramname NMTOKEN #IMPLIED
xml:id ID #IMPLIED
>
<!-- ELEMENT behavior EMPTY -->
<!-- ATTLIST behavior
    set NMTOKEN #REQUIRED
    set.paramname NMTOKEN #IMPLIED
    name NMTOKEN #REQUIRED
    name.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
-->
<!-- ELEMENT value (#PCDATA) -->
<!-- ATTLIST value
    value.paramname NMTOKEN #IMPLIED
    type (uint | sint | float | string | object) #REQUIRED
    type.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
-->
<!-- ELEMENT propertypointer EMPTY -->
<!-- ATTLIST propertypointer
    ref CDATA #REQUIRED
    ref.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
-->
<!-- included devices and parameter instantiation -->
<!-- ELEMENT includeddev (label?, protocol*, setparam*) -->
<!-- ATTLIST includeddev
    UUID NMTOKEN #REQUIRED
    UUID.paramname NMTOKEN #IMPLIED
    array CDATA #IMPLIED
    array.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
-->
<!-- ELEMENT setparam (#PCDATA) -->
<!-- ATTLIST setparam
    name NMTOKEN #REQUIRED
    setparam.paramname NMTOKEN #IMPLIED
    xml:id ID #IMPLIED
-->
<!-- Protocol dependent section -->
<!--
    useprotocol fixed for DMP
-->
<!-- ELEMENT useprotocol EMPTY -->
<!-- ATTLIST useprotocol
    name ( ESTA.DMP ) #REQUIRED
    xml:id ID #IMPLIED
-->

```

ANSI E1.17-2010, ACN – Device Description Language (DDL)

```
>
<!ELEMENT protocol ((propref_DMP?, childrule_DMP?) | propmap_DMx?) >
<!-- ATTLIST protocol
  name ( ESTA.DMP ) #REQUIRED
  xml:id ID #IMPLIED
-->
<!-- ELEMENT propref_DMP EMPTY -->
<!-- ATTLIST propref_DMP
  loc CDATA #REQUIRED
  loc.paramname NMTOKEN #IMPLIED
  abs (true | false) "false"
  abs.paramname NMTOKEN #IMPLIED
  inc CDATA #IMPLIED
  inc.paramname NMTOKEN #IMPLIED
  size CDATA #REQUIRED
  size.paramname NMTOKEN #IMPLIED
  read (true | false) "false"
  read.paramname NMTOKEN #IMPLIED
  write (true | false) "false"
  write.paramname NMTOKEN #IMPLIED
  event (true | false) "false"
  event.paramname NMTOKEN #IMPLIED
  varsize (true | false) "false"
  varsize.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
-->
<!-- ELEMENT childrule_DMP EMPTY -->
<!-- ATTLIST childrule_DMP
  loc CDATA #REQUIRED
  loc.paramname NMTOKEN #IMPLIED
  abs (true | false) "false"
  abs.paramname NMTOKEN #IMPLIED
  inc CDATA #IMPLIED
  inc.paramname NMTOKEN #IMPLIED
  xml:id ID #IMPLIED
-->
<!-- End of protocol dependent section -->
```

Appendix D. Languagesets and Behaviors Defined for DDL

A base set of property behaviors is defined. These and their associated strings are specified in [acnbaseDDL]. These are a normative part of the specification of DDL for DMP. However, they are best read using XML presentation tools (for example a browser in conjunction with a stylesheet, by transformation into HTML using XSLT or using a suitable XML editor).

Although these behaviors are defined as part of DDL-DMP, they may be freely referenced by DDL using other protocols without change of UUID provided that they do not conflict with rules specific to those other protocols.

Appendix E. Definitions

access protocol

The network or datalink protocol used to access and control the devices described by a Device Description (e.g. DMP for a DMP device [DMP]). DDL may be adapted to many different access protocols both within and outside of the ACN suite, and a single device may support multiple access protocols (see: The Access Protocol).

appliance

In DDL an appliance is a piece of equipment described by a root device and all its children and descendents. In DMP systems [DMP] an appliance corresponds to a component that exposes one or more devices (since the rules require that all devices are descendants of a single [root device](#)).

CID

Short for Component Identifier. The ACN Architecture [Arch] requires that each [component](#) has a unique identifying CID which is a 128-bit Universally Unique Identifier [UUID].

component

The process, program or application corresponding to a single ACN endpoint. All messages in ACN are sent and received by a component that is identified by a [CID](#). See [Arch] for a more complete definition.

controller

The term controller is often used loosely to refer to any piece of equipment that controls or monitors other equipment via the network. However, in the context of DMP a controller is defined precisely in terms of the messages implemented, while in DDL context a controller is defined by its use of device descriptions. Other protocols or contexts may have their own definitions. See [controller \(DDL\)](#).

controller (DDL)

Within DDL a controller is a network entity that interprets the DDL descriptions of devices to know how to access or control them using the [access protocol\(s\)](#) of the Device Description to access each device.

DCID

Device Class Identifier. This is a unique identifier [UUID] for a Device Class. All devices have a DCID that they share with all other devices of their class. The DCID may be used as a key to recognize or retrieve the device description for a device class.

device (DDL)

Within DDL, a device is a DDL module describing an entity that may be monitored and controlled by means of a network or datalink. In DDL there is no distinction

ANSI E1.17-2010, ACN – Device Description Language (DDL)

between a device and a sub-device except for the context in which they are encountered (device is a recursive term).

device class

The set of devices that are all described by the same device description.

module

In DDL elements [behaviorset](#), [device](#) and [languageset](#) are defined to be modules. Each module must be wrapped by itself in a DDL element which constitutes a XML document. See sections 3.3 , 4.4 , 4.5 , 4.6 .

root device

An instance of a device described in DDL that has no parent device. See also [appliance](#).

semantic

A term common in linguistics, computer science, logic and formal languages (e.g. DDL) that identifies the meaning of an item or construct as distinct from its syntax (the term grammar is loose and is sometimes taken to include semantics but more often excludes it).

Appendix F. References

Normative

[ACN] Entertainment Services and Technology Association, since 1 January 2011 PLASA North America [<http://tsp.plasa.org/>]. E1.17. Entertainment Technology – Architecture for Control Networks. The edition current when this Standard is approved.

[Arch] Entertainment Services and Technology Association, since 1 January 2011 PLASA North America [<http://tsp.plasa.org/>]. E1.17. Entertainment Technology – Architecture for Control Networks. “ACN” Architecture. The edition current when this Standard is approved..

[DDL] Entertainment Services and Technology Association, since 1 January 2011 PLASA North America [<http://tsp.plasa.org/>]. E1.17. Entertainment Technology – Architecture for Control Networks. Device Description Language. The edition current when this Standard is approved..

[DMP] Entertainment Services and Technology Association, since 1 January 2011 PLASA North America [<http://tsp.plasa.org/>]. E1.17. Entertainment Technology – Architecture for Control Networks. Device Management Protocol. The edition current when this Standard is approved..

[acnbaseDDL] Entertainment Services and Technology Association, since 1 January 2011 PLASA North America [<http://tsp.plasa.org/>]. urn:uuid:71576eac-e94a-11dc-b664-0017316c497d. ACN Base Behavior Set. As extended by the addition of behaviorsets that are identified as “Core Modules” according to [CoreDDL] when this Standard is approved..

[ESTA-IDs] Entertainment Services and Technology Association, since 1 January 2011 PLASA North America [<http://tsp.plasa.org/>]. E1.17. Entertainment Technology – Architecture for Control Networks. EPI-16 ESTA Registered Names and Identifiers – Format and Procedure for Registration. The edition current when this Standard is approved..

[CoreDDL] Entertainment Services and Technology Association, since 1 January 2011 PLASA North America [<http://tsp.plasa.org/>]. E1.17. Entertainment Technology – Architecture for Control Networks. EPI 22 Revised. DDL Core Modules for ACN Devices (Draft Standard). The edition current when this Standard is approved..

[ISO-DATE] International Standards Organisation [<http://www.iso.org/>]. ISO 8601. Data elements and interchange formats – Information interchange. Representation of dates and times. 2000.

[RELAXNG] Organization for the Advancement of Structured Information Standards (OASIS) [<http://www.oasis-open.org/>]. RELAX NG Specification. Committee Specification. 3 December 2001. <http://relaxng.org/spec-20011203.html>.

[LANG-TAG] Internet Engineering Task Force (IETF) [<http://ietf.org/>]. RFC 3066 [<http://ietf.org/rfc/rfc3066.txt>]. Alvestrand. Tags for the Identification of Languages. 2001.

[Unicode] The Unicode Consortium [<http://www.unicode.org/consortium/consort.html>]. The Unicode Standard [<http://www.unicode.org/versions/Unicode5.0.0/>], Version 5.0.0, defined by: The Unicode Standard, Version 5.0 (Boston, MA, Addison-Wesley, 2007. ISBN 0-321-48091-0)

[US-ASCII] American National Standards Institute [<http://www.ansi.org/>]. ANSI INCITS 4-1986 (R2007). Coded Character Sets – 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII). 2007.

[UUID] Internet Engineering Task Force (IETF) [<http://ietf.org/>]. RFC 4122 [<http://ietf.org/rfc/rfc4122.txt>]. P. Leach, M. Mealling, and R. Salz. A Universally Unique IDentifier (UUID) URN Namespace. July 2005.

ANSI E1.17-2010, ACN – Device Description Language (DDL)

[XML] World Wide Web Consortium (W3C) [<http://www.w3c.org/>]. Extensible Markup Language (XML) 1.0 (The edition current when this Standard is approved.). W3C Recommendation. [<http://www.w3.org/TR/xml/>].

[XMLnames] World Wide Web Consortium (W3C) [<http://www.w3c.org/>]. Namespaces in XML. W3C Recommendation. 14 January 1999. <http://www.w3.org/TR/REC-xml-names/>.

[XSL] World Wide Web Consortium (W3C) [<http://www.w3c.org/>]. Extensible Stylesheet Language (XSL). W3C Recommendation. 15 October 2001. <http://www.w3.org/TR/xsl>.

[XSD] World Wide Web Consortium (W3C) [<http://www.w3c.org/>]. XML Schema Part 2: Datatypes. W3C Recommendation. 28 October 2004. <http://www.w3.org/TR/xmlschema-2/>.

[xml-id] World Wide Web Consortium (W3C) [<http://www.w3c.org/>]. xml:id Version 1.0. W3C Recommendation. 9 September 2005. <http://www.w3.org/TR/xml-id/>.

Informative

[DMX512] Entertainment Services and Technology Association, since 1 January 2011 PLASA North America [<http://tsp.plasa.org/>]. ANSI E1.11. USITT DMX512-A – Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories. The edition current when this Standard is approved..

[RDM] Entertainment Services and Technology Association , since 1 January 2011 PLASA North America [<http://tsp.plasa.org/>]. ANSI E1.20. Entertainment Technology – Remote Device Management over DMX512 Networks. The edition current when this Standard is approved..

[XInclude] World Wide Web Consortium (W3C) [<http://www.w3c.org/>]. XML Inclusions (XInclude) Version 1.0. W3C Recommendation. 15 November 2006. <http://www.w3.org/TR/xinclude/>.

[XPointer] World Wide Web Consortium (W3C) [<http://www.w3c.org/>]. XML Pointer Framework. W3C Recommendation. 25 March 2003. <http://www.w3.org/TR/xptr-framework/>.

[XPath] World Wide Web Consortium (W3C) [<http://www.w3c.org/>]. XML Path Language (XPath). W3C Recommendation. 16 November 1999. <http://www.w3.org/TR/xpath>.

Entertainment Services and Technology Association



American National Standard E1.20 - 2006 Entertainment Technology RDM Remote Device Management Over DMX512 Networks

Entertainment Services and Technology Association



American National Standard E1.20 - 2006 Entertainment Technology RDM Remote Device Management Over DMX512 Networks CP/2003-1003r4

This edition of ANSI E1.20 was approved by American National Standards Institute on March 31, 2006.

©2006 ASC E1, Safety and Compatibility of Entertainment Technical Equipment and Practices, and its secretariat the Entertainment Services and Technology Association. All rights reserved. No part of this publication may be reproduced in any material form (including photocopying or storing by electronic means) without the written permission of the copyright holder. Any parties wishing to translate and publish this document in another language must receive permission from the copyright holder.

Notice and Disclaimer

ESTA and ANSI Accredited Standards Committee E1 (for which ESTA serves as the secretariat) do 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, or an American National Standard developed under Accredited Standards Committee E1 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 or Accredited Standards Committee E1.

ESTA and ANSI Accredited Standards Committee E1 (ASC E1) neither guaranty nor warrant 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 and ASC E1 do 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 judgement or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstance.

NOTE - The user's attention is called to the possibility that compliance with this standard may require use of an invention covered by patent rights.

By publication of this standard, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. The patent holder has, however, filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license.

Published By:

**Entertainment Services and
Technology Association**

875 Sixth Avenue, Suite 1005
New York, NY 10001 USA
Phone: +1-212-244-1505
Fax: +1-212-244-1502
Email: standards@esta.org

For Electronic Copies of this Document Contact:
ANSI

www.estafoundation.org

For Additional Copies of this Document Contact:

ESTA Publications

The ESTA Foundation
875 Sixth Avenue, Suite 1005
New York, NY 10001 USA
Phone: +1-212-244-1505
Fax: +1-212-244-1502

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, PLASA, 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 as Accredited Standards Committee E1, Safety and Compatibility of Entertainment Technical Equipment and Practices.

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

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 two 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 manufacturers, consultants, dealers, and end-users. ESTA is committed to developing consensus-based standards and recommended practices in an open setting. Future Control Protocols Working Group projects will include updating this publication as changes in technology and experience warrant, as well as developing new standards and recommended practices for the benefit of the entertainment industry.

Table of Contents

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	11
3.2.3 Responder discovery response driver enable time	11
3.2.4 Driver shutoff time	11
3.3 Data collisions	11
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	12
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	53
10.4.1 Get Supported Parameters (SUPPORTED_PARAMETERS)	54
10.4.2 Get Parameter Description (PARAMETER_DESCRIPTION)	55
10.5 Product Information Messages	57
10.5.1 Get Device Info (DEVICE_INFO)	57
10.5.2 Get Product Detail ID List (PRODUCT_DETAIL_ID_LIST)	60
10.5.3 Get Device Model Description (DEVICE_MODEL_DESCRIPTION)	61
10.5.4 Get Manufacturer Label (MANUFACTURER_LABEL)	61
10.5.5 Get/Set Device Label (DEVICE_LABEL)	62
10.5.6 Get/Set Factory Defaults (FACTORY_DEFAULTS)	63
10.5.7 Get Language Capabilities (LANGUAGE_CAPABILITIES)	64
10.5.8 Get/Set Language (LANGUAGE)	64
10.5.9 Get Software Version Label (SOFTWARE_VERSION_LABEL)	66
10.5.10 Get Boot Software Version ID (BOOT_SOFTWARE_VERSION_ID)	67
10.5.11 Get Boot Software Version Label (BOOT_SOFTWARE_VERSION_LABEL)	68
10.6 DMX512 Setup Messages	68
10.6.1 Get/Set DMX512 Personality (DMX_PERSONALITY)	68

10.6.2 Get DMX512 Personality Description (DMX_PERSONALITY_DESCRIPTION)	70
10.6.3 Get/Set DMX512 Starting Address (DMX_START_ADDRESS)	71
10.6.4 Get Slot Info (SLOT_INFO)	72
10.6.5 Get Slot Description (SLOT_DESCRIPTION)	73
10.6.6 Get Default Slot Value (DEFAULT_SLOT_VALUE)	74
10.7 Sensor Parameter Messages	74
10.7.1 Get Sensor Definition (SENSOR_DEFINITION)	75
10.7.2 Get/Set Sensor (SENSOR_VALUE)	77
10.7.3 Record Sensors (RECORD_SENSORS)	79
10.8 Power/Lamp Setting Parameter Messages	80
10.8.1 Get/Set Device Hours (DEVICE_HOURS)	80
10.8.2 Get/Set Lamp Hours (LAMP_HOURS)	81
10.8.3 Get/Set Lamp Strikes (LAMP_STRIKES)	82
10.8.4 Get/Set Lamp State (LAMP_STATE)	83
10.8.5 Get/Set Lamp On Mode (LAMP_ON_MODE)	84
10.8.6 Get/Set Device Power Cycles (DEVICE_POWER_CYCLES)	85
10.9 Display Setting Parameter Messages	86
10.9.1 Get/Set Display Invert (DISPLAY_INVERT)	86
10.9.2 Get/Set Display Level (DISPLAY_LEVEL)	87
10.10 Device Configuration Parameter Messages	88
10.10.1 Get/Set Pan Invert (PAN_INVERT)	88
10.10.2 Get/Set Tilt Invert (TILT_INVERT)	89
10.10.3 Get/Set Pan/Tilt Swap (PAN_TILT_SWAP)	90
10.10.4 Get/Set Device Real-Time Clock (REAL_TIME_CLOCK)	91
10.11 Device Control Parameter Messages	92
10.11.1 Get/Set Identify Device (IDENTIFY_DEVICE)	92
10.11.2 Reset Device (RESET_DEVICE)	93
10.11.3 Get/Set Power State (POWER_STATE)	94
10.11.4 Get/Set Perform Self Test (PERFORM_SELFTEST)	95
10.11.5 Get Self Test Description (SELF_TEST_DESCRIPTION)	96
10.11.6 Capture Preset (CAPTURE_PRESET)	97
10.11.7 Get/Set Preset Playback (PRESET_PLAYBACK)	98
11 Operational Issues	100
11.1 Polling Intervals	100
12 Protocol Support Issues	100
Appendix A: Defined Parameters (Normative)	101
Appendix B: Status Message ID's (Normative)	118
Appendix C: Slot Info (Normative)	120
Appendix D: Definitions (Normative)	122

<i>Appendix E: Discovery Pseudo-Code Example (Informative)</i>	126
E.1 Find Devices Function Call	126
E.2 Find Devices Pseudo-Code	126
<i>Appendix F: Qualification tests for transmitter/receiver circuits used in RDM systems (Normative)</i>	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	131
F.5 Testing Responder Transmitters	132

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	92
Table A-1: Command Class Defines	101
Table A-2: Response Type Defines.....	101
Table A-3: RDM Categories/Parameter ID Defines.....	102
Table A-4: Status Type Defines	103
Table A-5: Product Category Defines.....	104
Table A-6: Product Detail Defines	106
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	111
Table A-11: Power State Defines	112
Table A-12: Sensor Type Defines	112
Table A-13: Sensor Unit Defines	113
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.*

ESTA

875 Sixth Avenue, Suite 1005
New York, NY 10001
+1-212-244-1505
<http://www.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

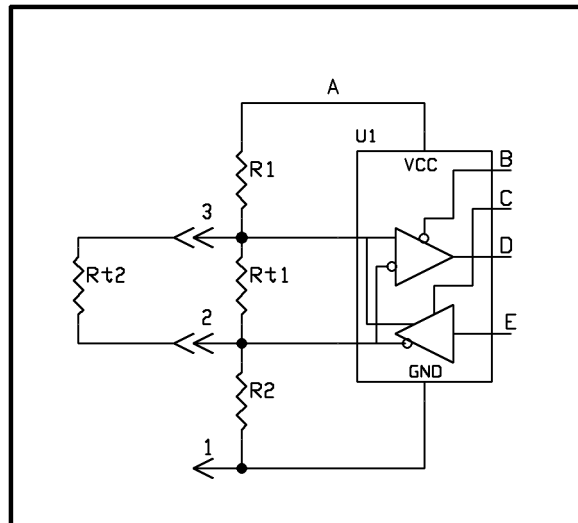


Figure 2-1: Command Port Reference Circuit

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 dependant 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
		<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>	<i>Max</i>
1	Transmit	176µS	352µS	12µS	88µS	0µS	2.0mS ^A	$440\mu\text{S}^{\text{B}} + (n^{\text{C}} \times 44\mu\text{S}^{\text{D}}) + ((n^{\text{C}} - 1) \times 76\mu\text{S}^{\text{E}})$
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.) $(44\mu\text{S} \times 24 \text{ bytes}) + (76\mu\text{S} \times 23) = 2.804\mu\text{S}$ 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
		<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>	<i>Max</i>
1	Receive	88µS	1S	8µS	1S	0µS	2.1mS	
2	Transmit	176µS	352µS	12µS	88µS	0µS	2.0mS ^A	$440\mu\text{S}^{\text{B}} + (n^{\text{C}} \times 44\mu\text{S}^{\text{D}}) + ((n^{\text{C}} - 1) \times 76\mu\text{S}^{\text{E}})$
3	Transmit Discovery Response	N/A	N/A	N/A	N/A	0uS	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 3 of Table 3-2 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

All responders shall enable their drivers no earlier than 4µS before the first start bit of a response to a discovery message. Prior to the beginning of the first start bit, the responder shall drive a MARK on the line.

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.

3.3 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 $\pm 1.875\%$ (75nS). This delay is not cumulative during a slot.

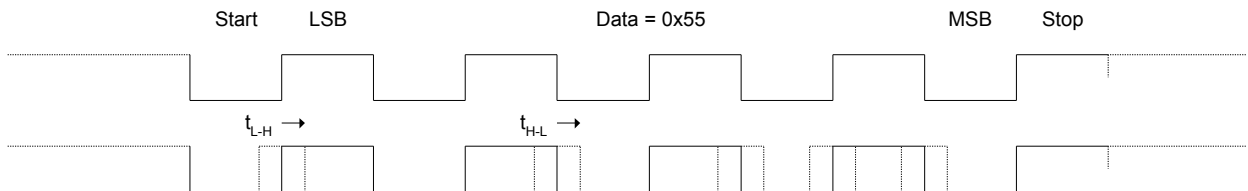


Figure 4-2: Bit Asymmetrical Delay

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	0	
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) 0x00 - 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) 0x00 - 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) 0x00 - 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.

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	0x02	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	0x00	
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	0xFF	STATUS_ERROR
25	Checksum High	0x07	0x0765
26	Checksum Low	0x65	

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) <div style="border: 1px solid black; padding: 2px; display: inline-block;">Estimated Response Time (16-bit)</div>		

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) 0x04
(PD)		
0x00000258		

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) 0x00 - 0xFF	(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.

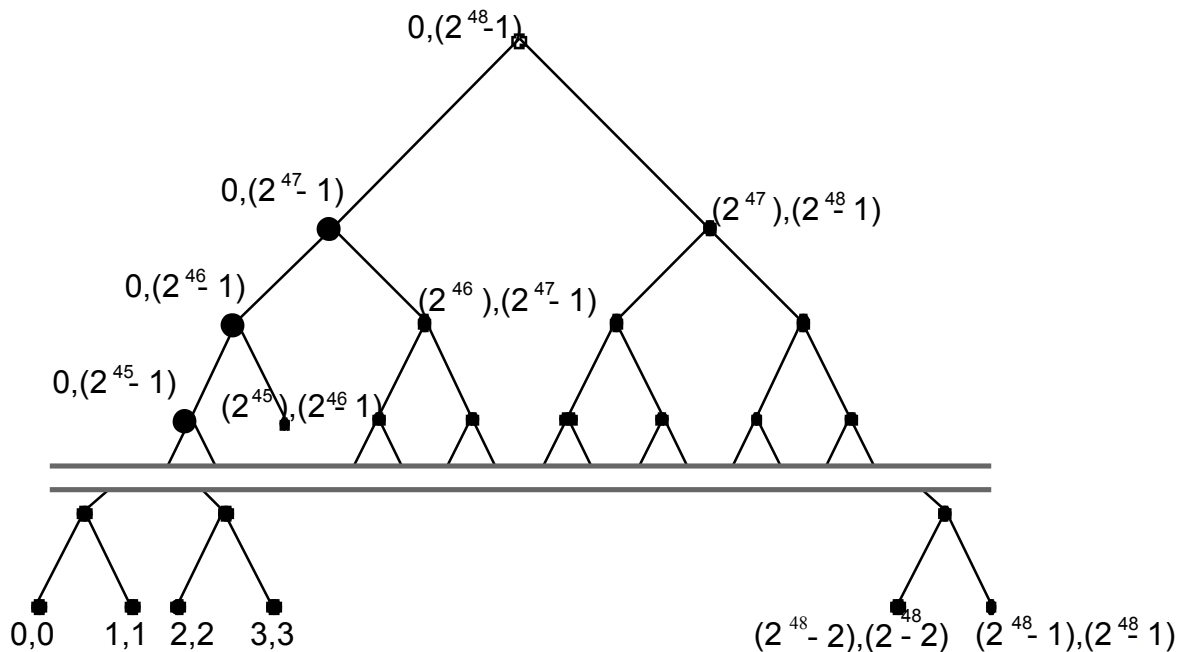


Figure 7-1: Binary Search

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) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000	
(CC) DISCOVERY_COMMAND	(PID) DISC_UNIQUE_BRANCH		(PDL) 0x0C
(PD)			
<div>Lower Bound UID (48-bit)</div> <div>Upper Bound UID (48-bit)</div>			

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	Bit 3	Bit 2	Bit 1	Bit 0
Reserved (Always set to 0)	Proxied Device Flag	Boot-Loader Flag	Sub-Device Flag	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.

Sub-Device Flag (Bit 1)

The Sub-Device Flag (Bit 1) shall be set to 1 when the responder supports Sub-Devices. See Section 9 for information on Sub-Devices.

Boot-Loader Flag (Bit 2)

The Boot-Loader Flag (Bit 2) shall only be set to 1 when the device is incapable of normal operation until receiving a firmware upload.

It is expected that when in this Boot-Loader mode the device will be capable of very limited RDM communication. The process of uploading firmware is beyond the scope of this document.

Proxied Device Flag (Bit 3)

The Proxied Device Flag (Bit 3) shall only be set to 1 when a Proxy is responding to Discovery on behalf of another device. This flag indicates that the response has come from a Proxy, rather than the actual device.

Reserved bits (Bits 4-15)

The Reserved bits (Bits 4-15) are reserved for future implementation and shall be set to 0.

7.6.2 Binding UID

The Binding UID field shall only be included when the responding device contains multiple responder ports. If the device does contain multiple ports, then the Binding UID field shall contain the UID for the primary port on the device.

This Binding UID field allows the controller to associate multiple responder ports discovered within a single physical device.

7.6.3 Discovery Mute Message (DISC_MUTE)

A responder port shall set its Mute flag when it receives this message containing its UID, or a broadcast address.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000
(CC) DISCOVERY_COMMAND	(PID) DISC_MUTE	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000
(CC) DISCOVERY _COMMAND_RESPONSE	(PID) DISC_MUTE	(PDL) 0x02 or 0x08
(PD)		
Control Field (16-bit)	Binding UID (Optional) (48-bit)	

7.6.4 Discovery Un-Mute Message (DISC_UN_MUTE)

A responder port shall clear its Mute flag when it receives this message containing its UID, or a broadcast address.

Controller:

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

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000
(CC) DISCOVERY _COMMAND_RESPONSE	(PID) DISC_UN_MUTE	(PDL) 0x02 or 0x08
(PD)		
Control Field (16-bit)	Binding UID (Optional) (48-bit)	

7.7 Discovery Algorithm

The discovery of devices will usually be accomplished using a binary-tree search algorithm. However, modifications of this method can be used and may be more appropriate in certain cases. The flowchart in Figure 7-2 illustrates the recursive portion of the binary search algorithm used to find undiscovered devices within a given UID range. Pseudo-code for this process is located in Appendix E.

Device discovery does not mandate discovery of all connected devices before sending other RDM commands to a discovered device.

A device need not be discovered and muted before accepting and properly acting on NSC packets or other RDM packets. That is, Discovery is not a mandatory requirement for proper device operation.

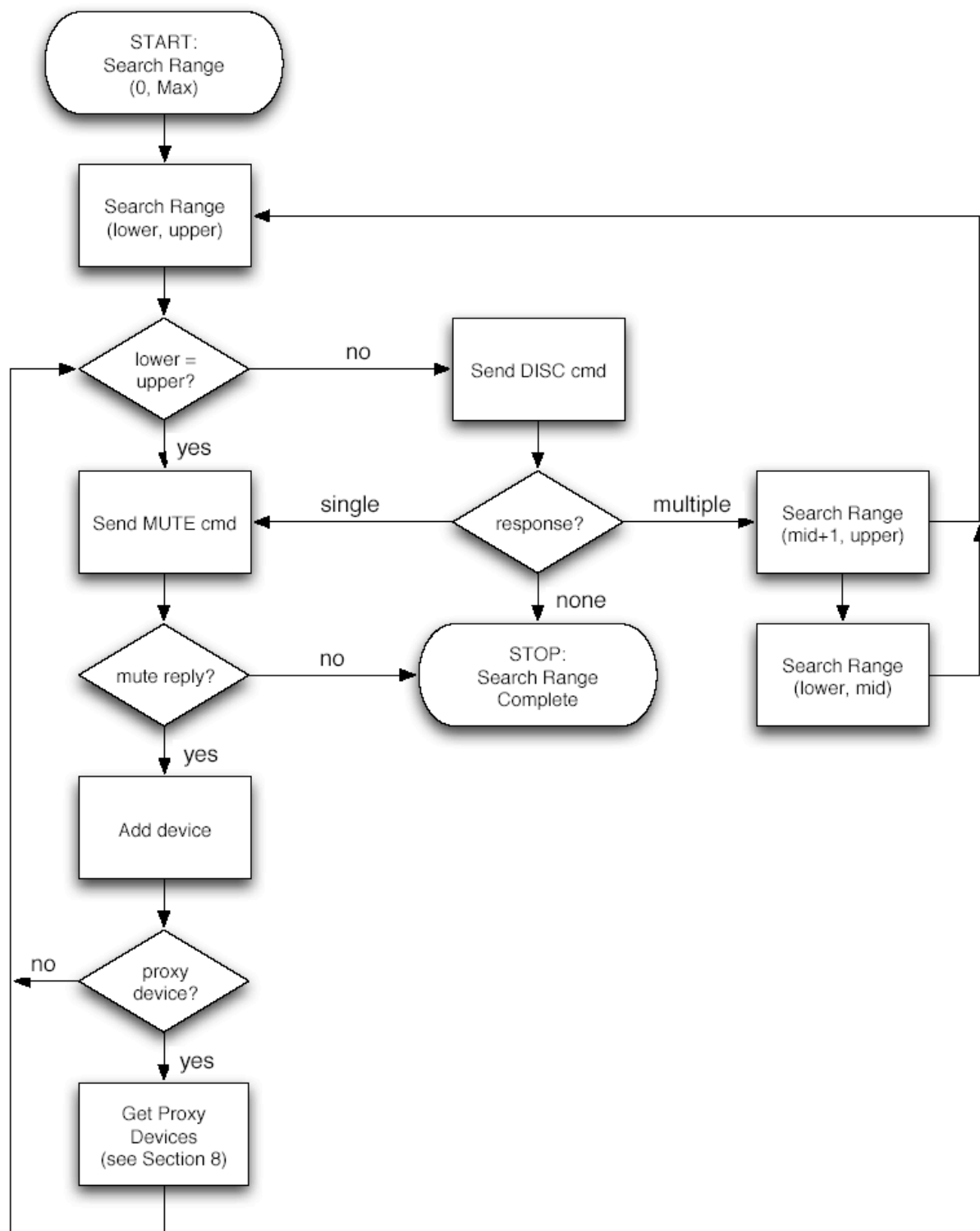


Figure 7-2: Device Discovery Process

7.8 Ongoing Device Discovery

The controller can detect any new devices that are added to the network by leaving discovered devices muted and sending a DISC_UNIQUE_BRANCH message covering the entire UID range. Only new devices (devices that are not muted) will respond.

Controllers may wish to periodically un-mute all devices then send individual mute commands to the devices it believes are connected. Following this with a DISC_UNIQUE_BRANCH message covering the entire UID range will detect devices that have been added. This technique will discover devices that were muted (for whatever reason) but that the controller did not know were connected.

8 Proxy Devices

8.1 General

A proxy is any in-line device that acts as an agent or representative for one or more devices. Represented devices may be non-RDM devices and may be logical (non-physical) devices.

A proxy shall respond to all controller messages on behalf of the devices it represents, as if it is the represented device.

A proxy itself may be discoverable as an RDM device, and may optionally support the Proxy Management commands which may be used by a controller to reduce the complexity of Discovery in a large system. A proxy with such capabilities is known as a managed proxy.

8.2 Discovery

All devices represented by a Proxy are discovered using the standard methods of Discovery described in Section 7.

8.2.1 Represented Devices

A proxy shall respond to all DISC_UNIQUE_BRANCH messages on behalf of each represented device as if it were the represented device. Each represented device should appear to the controller to be an independent RDM device.

A proxy shall provide no more than one response for a DISC_UNIQUE_BRANCH message.

A proxy device shall set the Proxied Device Flag to 1 when it is responding to the DISC_MUTE message on behalf of any represented device. This indicates that the response originates from the proxy rather than the actual device. The Managed Proxy Flag shall be cleared, as the response does not relate to the Proxy's own UID.

8.2.2 Proxy Management

A proxy device may itself be discoverable. If it is discoverable, then it shall set the Managed Proxy Flag to 1 when responding to its DISC_MUTE message. This indicates that the device is a Managed Proxy and may support the minimum set of Proxy Management commands.

A Managed Proxy device may support the PROXIED_DEVICES parameter in order to provide the controller with a list of UIDs for represented devices. A controller may choose to speed the discovery process by attempting to directly mute the undiscovered represented devices in the list.

8.3 Response Messages

A proxy may use the RESPONSE_TYPE_ACK_TIMER mechanism (See Section 6.3.3) when the controller requests information that is not immediately available from a represented device.

8.4 Proxy Management Messages

For discoverable proxies, the Proxy Management messages shall always be addressed to the Root Device of the Proxy.

8.4.1 Get Proxied Device Count (PROXIED_DEVICE_COUNT)

This parameter is used to identify the number of devices being represented by a proxy and whether the list of represented device UIDs has changed. If the List Change flag is set then the controller should GET: PROXIED_DEVICES.

The device shall automatically clear the List Change flag after all the proxied UID's have been retrieved using the GET: PROXIED_DEVICES message.

A proxy device shall indicate any change in it's device list through a QUEUED_MESSAGE for the PROXIED_DEVICE_COUNT PID. The controller may then get the current list of proxied devices by sending a GET_COMMAND for the PROXIED_DEVICES PID.

Controller: (GET)

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

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000		
(CC) GET_COMMAND_ RESPONSE	(PID) PROXIED_DEVICE_COUNT	(PDL) 0x03		
(PD)				
<table><tr><td>Device Count (16-bit)</td></tr><tr><td>List Change (0/1)</td></tr></table>			Device Count (16-bit)	List Change (0/1)
Device Count (16-bit)				
List Change (0/1)				

8.4.2 Get Proxied Devices (PROXIED_DEVICES)

This parameter is used to retrieve the UIDs from a device identified as a proxy during discovery. The response to this parameter contains a packed list of 48-bit UIDs for all devices represented by the proxy.

If there are no current devices being proxied then the Parameter Data Length field shall be returned as 0x00.

Controller:

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

Response:

(Response Type) ACK / ACK_OVERFLOW	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000
(CC) GET_COMMAND_ RESPONSE	(PID) PROXIED_DEVICES	(PDL) Variable (0x00 – 0xE4)
(PD)		
<div>Packed field with 48-bit UID's.</div>		

9 Sub-Devices

9.1 General

A device may contain one or more sub-devices, all having a common set of parameters. An example of a device with sub-devices is a dimmer rack containing a number of dimmer module sub-devices.

Sub-devices can only ever exist as part of a root device. Sub-devices may not themselves contain sub-devices.

9.2 Sub-Device Messages

9.2.1 Sub-Device Field

The Sub-Device field in the message structure provides a range of 512 sub-device offsets from 1 -512. The value of 0xFFFF is reserved for the SUB_DEVICE_ALL_CALL. A sub-device of 0x0000 is used to address the Root device or for devices that do not contain any sub-devices.

9.2.2 Using Sub-Devices

The Sub-Device Count field in the DEVICE_INFO PID shall be used to establish the number of sub-devices present in a responder. If this is non-zero, the controller may request the list of supported parameters from any one of the sub-devices.

Note that because this standard places no requirement that sub-devices be numbered contiguously, it may be necessary to scan the full sub-device range to locate the sub-devices.

Broadcast SET commands may be sent using the SUB_DEVICE_ALL_CALL Sub-Device ID.

9.2.3 Required Sub-Device Messages

Devices supporting the use of sub-devices shall support the SUPPORTED_PARAMETERS message in order for the controller to determine which additional messages are supported by the sub-devices.

All sub-devices shall report an identical list for SUPPORTED_PARAMETERS. This list may be different than that of the Root device. To obtain the list of Parameters supported by the sub-devices, a GET:SUPPORTED_PARAMETERS message must be sent to any of the valid sub-device addresses for the device.

10 RDM Parameter Messages

RDM Parameter Messages are sent to set configuration and get status information from the device.

10.1 Text Field Handling

A number of Parameter messages contain text description fields within them. These text fields shall use ASCII character encoding following the ISO/IEC 646 standard character set. Unless otherwise specified the text string length shall be variable up to 32 characters.

The Parameter Data Length shall accordingly be set to match the actual size of the text string being sent. Parsing of the string shall be length terminated corresponding to the Parameter Data Length field. Text fields shall terminate based on Parameter Data Length, however if a NULL is encountered then that shall also act as a terminator for the text field.

Controllers with limited display capabilities may be unable to display the complete text fields. In these cases the first 8 characters should be considered the most important.

10.2 Network Management Messages

Network Management Messages include messages used to determine the quality of the communication link and configuration of the network. Network Management messages shall always be addressed to Root Devices.

10.2.1 Communication Status (COMMS_STATUS)

The COMMS_STATUS parameter is used to collect information that may be useful in analyzing the integrity of the communication system.

A responder shall respond to a GET_COMMAND for this PID with a cumulative total of each error type in the response message defined below. Responders shall ignore any errors detected during the response period following a DISC_UNIQUE_BRANCH message, since it is expected that collisions may result in corrupted messages.

Controller: (GET)

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

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x00 (Root)			
(CC) GET_COMMAND_RESPONSE	(PID) COMMS_STATUS	(PDL) 0x06			
(PD)					
<table><tr><td>Short Message (16-bit)</td></tr><tr><td>Length Mismatch (16-bit)</td></tr><tr><td>Checksum Fail (16-bit)</td></tr></table>			Short Message (16-bit)	Length Mismatch (16-bit)	Checksum Fail (16-bit)
Short Message (16-bit)					
Length Mismatch (16-bit)					
Checksum Fail (16-bit)					

Short Message - The message ended before the Message Length field was received (see Section 6.2.3).

Length Mismatch - The number of slots actually received did not match the Message Length plus the size of the Checksum. This counter shall only be incremented if the Destination UID in the packet matches the Device's UID.

Checksum Fail - The message checksum failed (see Section 6.2.11). This counter shall only be incremented if the Destination UID in the packet matches the Device's UID.

A responder shall clear all of the error totals to zero when receiving a SET_COMMAND for this PID.

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x00 (Root)
(CC) SET_COMMAND	(PID) COMMS_STATUS	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x00 (Root)
(CC) SET_COMMAND_RESPONSE	(PID) COMMS_STATUS	(PDL) 0x00
(PD) Not Present		

10.3 Collection of Queued and Status Messages

Status Collection Messages include messages used to retrieve deferred (queued) responses, device status and error information, and information regarding the RDM parameters supported by the device. Status Collection messages are normally addressed to Root Devices.

10.3.1 Get Queued Message (QUEUED_MESSAGE)

The QUEUED_MESSAGE parameter shall be used to retrieve a message from the responder's message queue. The Message Count field of all response messages defines the number of messages that are queued in the responder. Each QUEUED_MESSAGE response shall be composed of a single message response.

A responder with multiple messages queued shall first respond with the most urgent message. The message count of the responder shall be decremented prior to sending the response.

A responder with no messages queued shall respond to a QUEUED_MESSAGE message with a STATUS_MESSAGES response. A STATUS_MESSAGES response with a PDL of 0x00 does not imply that the responder supports the STATUS_MESSAGES PID.

A SET_COMMAND has no action with QUEUED_MESSAGE.

Controller (GET):

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root)	
(CC) GET_COMMAND	(PID) QUEUED_MESSAGE		(PDL) 0x01
(PD)			
<div>Status Type Requested</div>			

The Status Type Requested is one of STATUS_GET_LAST_MESSAGE or as allowed in Table 10-1.

The Parameter Data field shall contain the Status Type Requested indicating that, if the response to the QUEUED_MESSAGE request is a STATUS_MESSAGES response, the returned information shall be as defined in Table 10-1.

If the Status Type Requested is STATUS_GET_LAST_MESSAGE, the responder shall return the last message (which may be either a Queued Message or a Status Message) sent in response to a GET: QUEUED_MESSAGE.

The following transaction shows an example of QUEUED_MESSAGE where the DMX512 Start Address has been locally changed at the device and a change in Device Hours has occurred.

Controller sends GET: QUEUED_MESSAGE:

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

Responder sends queued message for DMX_START_ADDRESS:

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

The Message Count in the response message indicates another Queued Message pending.
Controller sends another GET: QUEUED_MESSAGE.

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

Response message indicates a change in Device Hours status:

Response:

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

Controller sends GET: QUEUED_MESSAGE: (No Queued Messages Pending)

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

Response message returns a Status Message since there are no pending messages:

Response:

(Response Type) ACK	(Message Count) 0x00	(Sub-Device) 0x0000
(CC) GET_COMMAND_RESPONSE	(PID) STATUS_MESSAGES	(PDL) 0x12
(PD)		
<div> <div>Status Message # 1</div> <div> <div>Sub-Device ID (16-bit)</div> <div> <div>Status Type</div> <div>Status Message ID (16-bit)</div> <div>Data Value 1 (16-bit)</div> <div>Data Value 2 (16-bit)</div> </div> </div> <div>Status Message # 2</div> <div> <div>Sub-Device ID (16-bit)</div> <div> <div>Status Type</div> <div>Status Message ID (16-bit)</div> <div>Data Value 1 (16-bit)</div> <div>Data Value 2 (16-bit)</div> </div> </div> </div>		

10.3.2 Get Status Messages (STATUS_MESSAGES)

This parameter is used to collect Status or Error information from a device.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) STATUS_MESSAGES	(PDL) 0x01
(PD) <div>Status Type Requested</div>		

The Status Type STATUS_GET_LAST_MESSAGE shall be used by the Controller to request the retransmission of the last sent Status Message or Queued Message.

When requesting status messages from a device, the Status Messages in Table 10-1 shall be returned according to the Status Type Requested sent by the controller.

The Status Type of STATUS_NONE shall be used when a controller wants to establish whether a device is present on the network without retrieving any Status Message data from the device.

Response:

(Response Type) ACK / ACK_OVERFLOW	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000
(CC) GET_COMMAND_ RESPONSE	(PID) STATUS_MESSAGES	(PDL) Variable (0x00 – 0xE1)
(PD)		
Status Message # 1		
Sub-Device ID (16-bit)		
Status Type		
Status Message ID (16-bit)		
Data Value 1 (16-bit)		
Data Value 2 (16-bit)		
Status Message # 2		
Sub-Device ID (16-bit)		
Status Type		
Status Message ID (16-bit)		
Data Value 1 (16-bit)		
Data Value 2 (16-bit)		
Status Message # n		
Sub-Device ID (16-bit)		
Status Type		
Status Message ID (16-bit)		
Data Value 1 (16-bit)		
Data Value 2 (16-bit)		

The response is a packed list of all status messages for the device. The total number of messages is calculated by dividing the PDL by 9. Each status message shall be a fixed length of 9 bytes.

Due to the maximum packet length limitation, the total number of status messages sent within a single message cannot exceed 25.

The responder shall maintain reported status information until it has been successfully delivered to the controller. Status is considered successfully delivered when the responder receives a Status Type Requested other than STATUS_GET_LAST_MESSAGE.

The previously delivered status messages shall be cleared from the reporting queue once they have been successfully delivered to the Controller.

10.3.2.1 Sub-Device ID

In a system containing sub-devices, the Sub-Device field shall be used to indicate the sub-device to which the status message belongs. If the status message does not reference a particular sub-device the field shall be set to 0x0000, to reference the root device.

10.3.2.2 Status Type

The Status Type is used to identify the severity of the condition. The message shall be reported with a status type of: STATUS_ADVISORY, STATUS_WARNING, or STATUS_ERROR.

Table 10-1: Required Response to Status Requests

<i>Status Type Requested</i>	<i>Status Type Messages Returned</i>		
	ERROR	WARNING	ADVISORY
ERROR	X		
WARNING	X	X	
ADVISORY	X	X	X
NONE (not allowed for use with QUEUED_MESSAGE)			

All Status Types are enumerated in Table A-4.

10.3.2.3 Status Message ID

Status Message ID's within the range of 0x0000 — 0x7FFF are reserved for publicly defined Status Messages. More information on publicly defined Status Message ID's can be found in Appendix B.

Manufacturer Specific messages are in the range of 0x8000 — 0xFFDF. Each Manufacturer-specific Status ID shall have a unique meaning, which shall be consistent across all products having a given Manufacturer ID.

10.3.2.4 Data Value 1 and 2

Each Status Message supports the return of two separate data values relevant to the context of the specific message. The data value for ESTA public status messages is used to identify a property within the device to which the message corresponds. Each Data Value shall be a signed integer.

For Manufacturer-specific messages, each Data Value field shall only represent a single parameter. It shall not be bit-encoded to represent multiple parameters.

Status ID's not using the Data Value fields shall set the fields with 0x0000.

10.3.3 Get Status ID Description (STATUS_ID_DESCRIPTION)

This parameter is used to request an ASCII text description of a given Status ID. The description may be up to 32 characters.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root)	
(CC) GET_COMMAND	(PID) STATUS_ID_DESCRIPTION		(PDL) 0x02
(PD)			
Status ID being Requested (16-bit)			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000	
(CC) GET_COMMAND_ RESPONSE	(PID) STATUS_ID_DESCRIPTION		(PDL) 0 – 32 Number of characters being sent
(PD)			
ASCII Text Field of variable size.			

Refer to Appendix B for details on response string formatting and parsing.

10.3.4 Clear Status ID (CLEAR_STATUS_ID)

This parameter is used to clear the status message queue.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) SET_COMMAND	(PID) CLEAR_STATUS_ID		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) SET_COMMAND_RESPONSE	(PID) CLEAR_STATUS_ID		(PDL) 0x00
(PD) Not Present			

10.3.5 Get/Set Sub-Device Status Reporting Threshold (SUB_DEVICE_STATUS_REPORT_THRESHOLD)

This parameter is used to set the verbosity of Sub-Device reporting using the Status Type codes as enumerated in Table A-4 .

This feature is used to inhibit reports from, for example, a specific dimmer in a rack that is generating repeated errors.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0001 – 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) SUB_DEVICE_STATUS_REPORT_THRESHOLD	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0001 – 0x0200 or 0xFFFF
(CC) GET_COMMAND_RESPONSE	(PID) SUB_DEVICE_STATUS_REPORT_THRESHOLD	(PDL) 0x01
(PD)		
<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">Status Type</div>		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0001 – 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) SUB_DEVICE_STATUS_REPORT_THRESHOLD	(PDL) 0x01
(PD)		
<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">Status Type</div>		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0001 – 0x0200 or 0xFFFF
(CC) SET_COMMAND_RESPONSE	(PID) SUB_DEVICE_STATUS_REPORT_THRESHOLD	(PDL) 0x00
(PD) Not Present		

10.4 RDM Information Messages

RDM Information Messages include messages used to determine the RDM capabilities of the devices connected to the network.

10.4.1 Get Supported Parameters (SUPPORTED_PARAMETERS)

The SUPPORTED_PARAMETERS message is used to retrieve a packed list of supported PIDs.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) – 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) SUPPORTED_PARAMETERS		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK / ACK_OVERFLOW	(Message Count) 0x00-0xFF	(Sub-Device) 0x00 – 0x0200 or 0xFFFF	
(CC) GET_COMMAND_RESPONSE	(PID) SUPPORTED_PARAMETERS		(PDL) Variable (0x00 – 0xE6)
(PD) Packed List of 16-bit Parameter ID's supported			

All Sub-Devices of a single Root Device shall report identical parameter lists although all parameters need not be implemented in each sub-device.

A Supported Parameters request sent to Sub-Device 0x0000 (Root) shall return the Parameters supported by the Root-Device. A Supported Parameters request sent to any other Sub-Device value (0x0001-0x0200) shall have an identical response as any other non-Root value Sub-Device. Thus, it is only required to request the Supported Parameters of the Root Device and a single Sub-Device, if Sub-Devices are supported.

Devices with the same Manufacturer ID, Device Model ID, and Software Version ID response for the DEVICE_INFO parameter shall report an identical list for the SUPPORTED_PARAMETERS message. This allows the controller to reduce the quantity of information stored for identical devices.

Manufacturer-Specific PIDs may or may not be included in the response.

PIDs that are included in the minimum support list, indicated by the “Required” column of Table A-3, shall not be reported.

10.4.2 Get Parameter Description (PARAMETER_DESCRIPTION)

This parameter is used to retrieve the definition of some manufacturer-specific PIDs. The purpose of this parameter is to allow a controller to retrieve enough information about the manufacturer-specific PID to generate Get and Set commands.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root)
(CC) GET_COMMAND	(PID) PARAMETER_DESCRIPTION	(PDL) 0x02
(PD) PID # Requested		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000
(CC) GET_COMMAND_RESPONSE	(PID) PARAMETER_DESCRIPTION	(PDL) 0x14 – 0x34
(PD)		
PID # Requested (16-bit)		
PDL Size		
Data Type		
Command Class		
Type		
Unit		
Prefix		
Min Valid Value (32-bit)		
Max Valid Value (32-bit)		
Default Value (32-bit)		
Description [Variable 0-32 Chars]		

Data Description:

PID # Requested:

The manufacturer specific PID requested by the controller. Range 0x8000 to 0xFFDF.

PDL Size:

PDL Size defines the number used for the PDL field in all GET_RESPONSE and SET messages associated with this PID. In the case of the value of DS_ASCII, the PDL Size represents the maximum length of a variable sized ASCII string.

Data Type:

Data Type defines the size of the data entries in the PD of the message for this PID. For example: unsigned 8-bit character versus signed 16-bit word. Table A-15 enumerates the field codes.

Command Class:

Command Class defines whether Get and or Set messages are implemented for the specified PID. Table A-16 enumerates the field codes.

Type:

Type is an unsigned 8-bit value enumerated by Table A-12. It defines the type of data that is described by the specified PID.

Unit:

Unit is an unsigned 8-bit value enumerated by Table A-13. It defines the SI (International System of units) unit of the specified PID data.

Prefix:

Prefix is an unsigned 8-bit value enumerated by Table A-14. It defines the SI Prefix and multiplication factor of the units.

Min Valid Value:

This is a 32-bit field that represents the lowest value that data can reach. The format of the number is defined by DATA TYPE. This field has no meaning for a Data Type of DS_BIT_FIELD or DS_ASCII. For Data Types less than 32-bits, the Most Significant Bytes shall be padded with 0x00 out to 32-bits. For example, an 8-bit data value of 0x12 shall be represented in the field as: 0x00000012.

Max Valid Value:

This is a 32-bit field that represents the highest value that data can reach. The format of the number is defined by DATA TYPE. This field has no meaning for a Data Type of DS_BIT_FIELD or DS_ASCII. For Data Types less than 32-bits, the Most Significant Bytes shall be padded with 0x00 out to 32-bits. For example, an 8-bit data value of 0x12 shall be represented in the field as: 0x00000012.

Default Value:

This is a 32-bit field that represents the default value of that data. This field has no meaning for a Data Type of DS_BIT_FIELD or DS_ASCII. The default value shall be within the minimum and maximum range. For Data Types less than 32-bits, the Most Significant Bytes shall be padded with 0x00 out to 32-bits. For example, an 8-bit data value of 0x12 shall be represented in the field as: 0x00000012.

Description:

The Description field is used to describe the function of the specified PID. This text field shall be variable up to 32 characters in length.

10.5 Product Information Messages

10.5.1 Get Device Info (DEVICE_INFO)

This parameter is used to retrieve a variety of information about the device that is normally required by a controller.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) DEVICE_INFO		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF											
(CC) GET_COMMAND_RESPONSE	(PID) DEVICE_INFO		(PDL) 0x13										
(PD)													
<table><tr><td>RDM Protocol Version (16-bit)</td></tr><tr><td>Device Model ID (16-bit)</td></tr><tr><td>Product Category (16-bit)</td></tr><tr><td>Software Version ID (32-bit)</td></tr><tr><td>-----</td></tr><tr><td>DMX512 Footprint (16-bit)</td></tr><tr><td>DMX512 Personality (16-bit)</td></tr><tr><td>DMX512 Start Address (16-bit)</td></tr><tr><td>Sub-Device Count (16-bit)</td></tr><tr><td>Sensor Count</td></tr></table>				RDM Protocol Version (16-bit)	Device Model ID (16-bit)	Product Category (16-bit)	Software Version ID (32-bit)	-----	DMX512 Footprint (16-bit)	DMX512 Personality (16-bit)	DMX512 Start Address (16-bit)	Sub-Device Count (16-bit)	Sensor Count
RDM Protocol Version (16-bit)													
Device Model ID (16-bit)													
Product Category (16-bit)													
Software Version ID (32-bit)													

DMX512 Footprint (16-bit)													
DMX512 Personality (16-bit)													
DMX512 Start Address (16-bit)													
Sub-Device Count (16-bit)													
Sensor Count													

Data Description:

RDM Protocol Version:

This field contains the version number of the published RDM Standard supported by the device. The value for these version fields are controlled by this standard and any subsequent revisions or additions to this standard as issued in the future.

The version of this standard is 1.0. The values for this version field shall only be changed by future versions, revisions or addendums to this document. The 16-bit field is encoded into 8-bit Major and Minor versions as shown below.

Major Version Release (0x01)	Minor Version Release (0x00)
---------------------------------	---------------------------------

The response for this field shall always be same regardless of whether this message is directed to the Root Device or a Sub-Device.

Device Model ID:

This field identifies the Device Model ID of the Root Device or the Sub-Device. The Manufacturer shall not use the same ID to represent more than one unique model type.

A text description for the Device Model ID can be retrieved from the device using the DEVICE_MODEL_DESCRIPTION Parameter.

Product Category:

Devices shall report a Product Category based on the product's primary function.

Product Categories are encoded as 16-bit values as defined in Table A-5. These are arranged so that the upper eight bits defines a coarse product categorization, and the lower eight bits additional (optional) fine categorization as shown below.

Product Category Coarse	Product Category Fine
----------------------------	--------------------------

Manufacturers are also encouraged to declare and support Product Detail in accordance with Section 10.5.2 and Table A-6.

Software Version ID:

This field indicates the Software Version ID for the device. The Software Version ID is a 32-bit value determined by the Manufacturer.

The 32-bit Software Version ID may use any encoding scheme such that the Controller may identify devices containing the same software versions.

Any devices from the same manufacturer with differing software shall not report back the same Software Version ID.

A meaningful description of the software version for display to the user may be obtained by using the SOFTWARE_VERSION_LABEL Parameter in Section 10.5.9.

DMX512 Footprint:

This field specifies the DMX512 footprint (number of consecutive DMX512 slots required). This information can be used in conjunction with DMX_ADDRESS for auto-patching capabilities.

If the DEVICE_INFO message is directed to a Sub-Device, then the response for this field contains the DMX512 Footprint for that Sub-Device. If the message is sent to the Root Device, it is the Footprint for the Root Device itself. If the Device or Sub-Device does not utilize Null Start Code packets for any control or functionality then it shall report a Footprint of 0x0000.

The range for this field is from 0 – 512.

DMX512 Personality:

The DMX512 Personality fields are detailed in Section 10.6.1. The 16-bit field is encoded into two 8-bit fields as shown below.

Current Personality	Total # of Personalities
---------------------	--------------------------

DMX512 Start Address:

The DMX512 Start Address field is detailed in Section 10.6.3.

If the Device or Sub-Device that this message is directed to has a DMX512 Footprint of 0, then this field shall be set to 0xFFFF.

Sub-Device Count:

This parameter is used to retrieve the number of Sub-Devices represented by the Root Device as described in Section 9.

The response for this field shall always be same regardless of whether this message is directed to the Root Device or a Sub-Device.

Sensor Count:

This field indicates the number of available sensors in a Root Device or Sub-Device. When this parameter is directed to a Sub-Device, the reply shall be identical for any Sub-Device owned by a specific Root Device.

When a device or sub-device is fitted with a single sensor, it would return a value of 0x01 for the Sensor Count. This sensor would then be addressed as Sensor Number 0x00 when using the other sensor-related parameter messages.

10.5.2 Get Product Detail ID List (PRODUCT_DETAIL_ID_LIST)

This parameter shall be used for requesting technology details for a device. The response is a packed message containing up to six product detail identifications. Product Detail ID's are defined in Table A-6.

Product details supplement the Product Category obtained using the DEVICE_INFO parameter as described in Section 10.5.1.

Product Detail information may be used by the controller for grouping or other sorting methods when patching or displaying system information. Not all Product Detail definitions may be appropriate for all Product Categories.

This standard does not place any restrictions on the use of Product Categories and Product Detail, which are intended to convey general information about the product to the controller.

Devices fitted with Residual Current Detectors (RCD) or Ground Fault Interrupt (GFI) devices may declare such functionality using this PID.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) PRODUCT_DETAIL_ID_LIST		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK / ACK_OVERFLOW	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_RESPONSE	(PID) PRODUCT_DETAIL_ID_LIST		(PDL) 0x02 – 0x0C
(PD) Packed list of 16-bit Product Detail ID's limited to six entries maximum			

10.5.3 Get Device Model Description (DEVICE_MODEL_DESCRIPTION)

This parameter provides a text description of up to 32 characters for the device model type.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) DEVICE_MODEL_DESCRIPTION	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) DEVICE_MODEL_DESCRIPTION	(PDL) 0-32 Variable
(PD) ASCII text Model description Up to 32 characters.		

10.5.4 Get Manufacturer Label (MANUFACTURER_LABEL)

This parameter provides an ASCII text response with the Manufacturer name for the device of up to 32 characters. The Manufacturer name must be consistent between all products manufactured within an ESTA Manufacturer ID.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) MANUFACTURER_LABEL	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) MANUFACTURER_LABEL	(PDL) 0-32 Variable
(PD) ASCII text Manufacturer description Up to 32 characters.		

10.5.5 Get/Set Device Label (DEVICE_LABEL)

This parameter provides a means of setting a descriptive label for each device. This may be used for identifying a dimmer rack number or specifying the device's location.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) DEVICE_LABEL	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) DEVICE_LABEL	(PDL) 0-32 Variable
(PD) <div>ASCII text label. Up to 32 characters.</div>		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) DEVICE_LABEL	(PDL) 0-32 Variable
(PD) <div>ASCII text label. Up to 32 characters.</div>		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) DEVICE_LABEL	(PDL) 0x00
(PD) Not Present		

10.5.6 Get/Set Factory Defaults (FACTORY_DEFAULTS)

This parameter is used to instruct a device to revert to its Factory Default user settings or configuration as determined by the Manufacturer.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) FACTORY_DEFAULTS	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) FACTORY_DEFAULTS	(PDL) 0x01
(PD) <div>True/False (1/0)</div>		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001 – 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) FACTORY_DEFAULTS	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) FACTORY_DEFAULTS	(PDL) 0x00
(PD) Not Present		

The Parameter Data value is used in the GET_COMMAND_RESPONSE message to indicate whether the device is currently set to the Factory Defaults.

10.5.7 Get Language Capabilities (LANGUAGE_CAPABILITIES)

This parameter is used to identify languages that the device supports for using the LANGUAGE parameter. The response contains a packed message of 2 character Language Codes as defined by ISO 639-1. International Standard ISO 639-1, Code for the representation of names of languages - Part 1: Alpha 2 code.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) LANGUAGE_CAPABILITIES		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK / ACK_OVERFLOW	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) LANGUAGE_CAPABILITIES		(PDL) Variable (0x00 – 0xE6)
(PD) Packed List of 2 character Language codes			

10.5.8 Get/Set Language (LANGUAGE)

This parameter is used to change the language of the messages from the device. Supported languages of the device can be determined by the LANGUAGE_CAPABILITIES.

The Language Codes are 2 character alpha codes as defined by ISO 639-1. International Standard ISO 639-1, Code for the representation of names of languages - Part 1: Alpha 2 code.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) LANGUAGE		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) LANGUAGE		(PDL) 0x02
(PD) <div>2 character alpha code for ISO 639-1</div>			

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) SET_COMMAND	(PID) LANGUAGE		(PDL) 0x02
(PD) <div>2 character alpha code for ISO 639-1</div>			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) SET_COMMAND_ RESPONSE	(PID) LANGUAGE		(PDL) 0x00
(PD) Not Present			

Set Language Code to 0x0000 to reset device to Default Language.

10.5.9 Get Software Version Label (SOFTWARE_VERSION_LABEL)

This parameter is used to get a descriptive ASCII text label for the device's operating software version. The descriptive text returned by this parameter is intended for display to the user. The label may be up to 32 characters.

The Software Version ID field from the DEVICE_INFO parameter should be used for comparing devices from the same Manufacturer with the same Device Model ID to determine if they are running identical software versions.

Controller:

(Port ID) 0x00 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) SOFTWARE_VERSION_LABEL		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) SOFTWARE_VERSION_LABEL		(PDL) 0 – 32 Variable
(PD) ASCII text Software Version Label Up to 32 characters.			

10.5.10 Get Boot Software Version ID (BOOT_SOFTWARE_VERSION_ID)

This parameter is used to retrieve the unique Boot Software Version ID for the device. The Boot Software Version ID is a 32-bit value determined by the Manufacturer.

The 32-bit Boot Software Version ID may use any encoding scheme such that the Controller may identify devices containing the same boot software versions.

Any devices from the same manufacturer with differing boot software shall not report back the same Boot Software Version ID.

A meaningful description of the boot software version for display to the user may be obtained by using the BOOT_SOFTWARE_VERSION_LABEL Parameter in Section 10.5.11.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) BOOT_SOFTWARE_VERSION_ID		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) BOOT_SOFTWARE_VERSION_ID		(PDL) 0x04
(PD) Boot Software Version ID (32-bit)			

10.5.11 Get Boot Software Version Label (BOOT_SOFTWARE_VERSION_LABEL)

This parameter is used to get a descriptive ASCII text label for the Boot Version of the software for Devices that support this functionality. The descriptive text returned by this parameter is intended for display to the user. The label may be up to 32 characters.

The BOOT_SOFTWARE_VERSION_ID parameter should be used for comparing devices from the same Manufacturer with the same Device Model ID to determine if they are running identical boot software versions.

Controller:

(Port ID) 0x00 – 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) BOOT_SOFTWARE_VERSION_LABEL		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) BOOT_SOFTWARE_VERSION_LABEL		(PDL) 0 – 32 Variable
(PD) ASCII text Software Boot Version Label Up to 32 characters.			

10.6 DMX512 Setup Messages

10.6.1 Get/Set DMX512 Personality (DMX_PERSONALITY)

This parameter is used to set the responder's DMX512 Personality. Many devices such as moving lights have different DMX512 "Personalities". Many RDM parameters may be affected by changing personality.

The DMX512 Personality can also be retrieved as part of the DEVICE_INFO Parameter Message in Section 10.5.1.

The GET_COMMAND_RESPONSE message includes the current DMX512 Personality setting and also the total number of personalities available. These personalities shall be consecutively numbered within the responder starting from 1.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) DMX_PERSONALITY	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) DMX_PERSONALITY	(PDL) 0x02	
(PD)			
Current Personality		# of Personalities	

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) SET_COMMAND	(PID) DMX_PERSONALITY		(PDL) 0x01
(PD)			
<div>Personality</div>			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) DMX_PERSONALITY	(PDL) 0x00
(PD) Not Present		

The DMX512 Slot Footprint needed can be accessed from the DMX512 Footprint field in the DEVICE_INFO Parameter. Text descriptions for a given personality can be retrieved using the DMX_PERSONALITY_DESCRIPTION Parameter.

10.6.2 Get DMX512 Personality Description (DMX_PERSONALITY_DESCRIPTION)

This parameter is used to get a descriptive ASCII text label for a given DMX512 Personality. The label may be up to 32 characters.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) DMX_PERSONALITY_DESCRIPTION	(PDL) 0x01
(PD) <div>Personality Requested</div>		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) DMX_PERSONALITY_DESCRIPTION	(PDL) 3 – 35 (3 + Number of characters sent)
(PD) <div><div>Personality Requested</div><div>DMX512 Slots Required 0-512 (16-bit)</div><div>ASCII Text Field of variable size</div></div>		

The Response data contains the Personality Requested, the DMX512 Footprint for that Personality, along with up to 32 characters of description.

10.6.3 Get/Set DMX512 Starting Address (DMX_START_ADDRESS)

This parameter is used to set or get the DMX512 start address.

The DMX512 Starting Address can also be retrieved as part of the DEVICE_INFO Parameter Message in Section 10.5.1.

The returned data represents the address in the range 1 to 512. A value of zero represents 'Not Set'. When this message is directed to a Root Device or Sub-Device that has a DMX512 Footprint of 0 for that Root or Sub-Device, then the response shall be set to 0xFFFF. Values outside this range are beyond the scope of this standard.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) DMX_START_ADDRESS		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) DMX_START_ADDRESS		(PDL) 0x02
(PD) DMX512 Address 1-512, 0xFFFF (16-bit)			

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) SET_COMMAND	(PID) DMX_START_ADDRESS		(PDL) 0x02
(PD) DMX512 Address 1-512 (16-bit)			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) SET_COMMAND_ RESPONSE	(PID) DMX_START_ADDRESS		(PDL) 0x00
(PD) Not Present			

10.6.4 Get Slot Info (SLOT_INFO)

This parameter is used to retrieve basic information about the functionality of the DMX512 slots used to control the device. The response is a packed list of Slot Label ID's referencing into the Slot Labels table. See Appendix C for more information on Slot Type and Slot Label ID's.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) SLOT_INFO	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK / ACK_OVERFLOW	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) SLOT_INFO	(PDL) Variable (0x00 – 0xE6)
(PD)		
Slot Offset 0		
0 (16-bit)		Slot Type
Slot Label ID (16-bit)		
Slot Offset 1		
1 (16-bit)		Slot Type
Slot Label ID (16-bit)		
Slot Offset n		
N (16-bit)		Slot Type
Slot Label ID (16-bit)		

The Slot Offset is a reference to the offset from the DMX512 Starting Address for the device or Sub-Device described by the Slot Label ID. The Slot Offset for the first slot of a device (i.e. the DMX512 Starting Address) is 0. If a device skips or does not use all slots within its footprint range, then those slots shall not be reported in this message.

The Slot Type field indicates whether the slot is primary or secondary. A primary type slot refers to the basic parameter, such as intensity or pan. The slots have a type of ST_PRIMARY, and the Slot ID/Offset field contains a Slot ID. A secondary type slot is an additional slot related to a primary parameter. For secondary types, the Slot ID/Offset field contains a slot offset for the parameter to which it relates. For example, a 16-bit pan function in slots 1 and 2 would use ST_PRIMARY/SD_PAN for slot 1 and ST_SEC_FINE/1 for slot 2. If a secondary slot relates to

more than one primary slot, it should reference the first applicable slot. Refer to Appendix C for more information on defined Slot Types and Slot IDs.

A fixture with multiple parameters of the same type should report multiple slots with the same code. For example, a fixture with two static gobo wheels would report two slots of ST_PRIMARY/SD_STATIC_GOBO_WHEEL, which a controller could then interpret as Static Gobo Wheel 1 and Static Gobo Wheel 2.

A Slot Label ID of 0xFFFF indicates that the description for that slot offset is only available by using the SLOT_DESCRIPTION parameter.

10.6.5 Get Slot Description (SLOT_DESCRIPTION)

This parameter is used for requesting an ASCII text description for DMX512 slot offsets.

If the country code of the device is set to the English language, then the text returned should be similar to the description as defined in Table C-2.

If the responder does not support the Slot number requested, or cannot provide a text description for a slot number that it does support, the responder shall respond with NR_DATA_OUT_OF_RANGE.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) SLOT_DESCRIPTION		(PDL) 0x02
(PD)			
Slot # Requested (16-bit)			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) SLOT_DESCRIPTION		(PDL) 2-34 (2+Number of characters being sent)
(PD)			
ASCII Text Field of variable size up to 32 characters			

10.6.6 Get Default Slot Value (DEFAULT_SLOT_VALUE)

This parameter shall be used for requesting the default values for the given DMX512 slot offsets for a device. The response is a packed message containing both the slot offset and its default value.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) DEFAULT_SLOT_VALUE	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK / ACK_OVERFLOW	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) DEFAULT_SLOT_VALUE	(PDL) Variable (0x00 – 0xE7)
(PD)		
Slot Offset 0		
0 (16-bit)		Default Slot Value
Slot Offset 1		
1 (16-bit)		Default Slot Value
Slot Offset n		
N (16-bit)		Default Slot Value

The Slot Offset is a reference to the offset from the DMX512 Starting Address for the device or Sub-Device described by the Slot Label ID. The Slot Offset for the first slot of a device (i.e. the DMX512 Starting Address) is 0. If a device skips or does not use all slots within its footprint range, then those slots shall not be reported in this message.

10.7 Sensor Parameter Messages

Responding devices may contain various sensors. The controller retrieves sensor data using this command subset.

All parameter messages within this section can be used with Sub-Devices. The Root Device may contain a maximum of 255 sensors. Sub-Devices may contain a maximum of 255 sensors. However, all Sub-Devices that are owned by a specific Root Device shall respond with an identical number of sensors.

Sensor messages may be addressed to Root Devices or Sub-Devices.

Valid sensor numbers are in the range from 0x00 – 0xFE. The sensor number 0xFF is used to address all sensors. The number of sensors present in a device can be retrieved using the DEVICE_INFO Parameter.

The number of sensors fitted to a responding device is obtained as part of the GET:DEVICE_INFO response.

10.7.1 Get Sensor Definition (SENSOR_DEFINITION)

This parameter is used to retrieve the definition of a specific sensor. When this parameter is directed to a Sub-Device, the reply shall be identical for any given sensor number in all Sub-Devices owned by a specific Root Device.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) SENSOR_DEFINITION		(PDL) 0x01
(PD)			
<div>Sensor # Requested</div>			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF											
(CC) GET_COMMAND_RESPONSE	(PID) SENSOR_DEFINITION		(PDL) 0x0D – 0x2D										
(PD)													
<table><tr><td>Sensor # Requested</td></tr><tr><td>Type</td></tr><tr><td>Unit</td></tr><tr><td>Prefix</td></tr><tr><td>Range Minimum Value (16-bit)</td></tr><tr><td>Range Maximum Value (16-bit)</td></tr><tr><td>Normal Minimum Value (16-bit)</td></tr><tr><td>Normal Maximum Value (16-bit)</td></tr><tr><td>Recorded Value Support</td></tr><tr><td>Description [Variable 0-32 chars]</td></tr></table>				Sensor # Requested	Type	Unit	Prefix	Range Minimum Value (16-bit)	Range Maximum Value (16-bit)	Normal Minimum Value (16-bit)	Normal Maximum Value (16-bit)	Recorded Value Support	Description [Variable 0-32 chars]
Sensor # Requested													
Type													
Unit													
Prefix													
Range Minimum Value (16-bit)													
Range Maximum Value (16-bit)													
Normal Minimum Value (16-bit)													
Normal Maximum Value (16-bit)													
Recorded Value Support													
Description [Variable 0-32 chars]													

Data Description:

Sensor Number:

The sensor number requested is in the range from 0x00 to 0xFE.

Type:

Type is an unsigned 8-bit value enumerated by Table A-12. It defines the type of data that is measured by the sensor.

Unit:

Unit is an unsigned 8-bit value enumerated by Table A-13. It defines the SI unit of the sensor data.

Prefix:

Prefix is an unsigned 8 bit value enumerated by Table A-14. It defines the SI Prefix and multiplication factor of the units.

Range Minimum Value:

This is a 2's compliment signed 16-bit value that represents the lowest value the sensor can report. A value of -32768 indicates that the minimum is not defined.

Range Maximum Value:

This is a 2's compliment signed 16-bit value that represents the highest value the sensor can report. This also defines the maximum capacity. A value of +32767 indicates that the maximum is not defined.

Normal Minimum Value:

This is a 2's compliment signed 16-bit value that defines the lowest sensor value for which the device is in normal operation. A value of -32768 indicates that the minimum is not defined.

Normal Maximum Value:

This is a 2's compliment signed 16-bit value that defines the highest value that for which the device is in normal operation. A value of +32767 indicates that the maximum is not defined.

Recorded Value Support:

This field is a bit-masked field to indicate the support for recorded values.

Bits 7-2	Bit 1	Bit 0
Reserved (Always set to 0)	Lowest/Highest Detected Values Supported	Recorded Value Supported

The Detected and Recorded Value fields are described in Section 10.7.2 Get/Set Sensor.

Description:

The Description field is used to describe the function of the specified Sensor. This text field shall be variable up to 32 characters in length.

10.7.2 Get/Set Sensor (SENSOR_VALUE)

This parameter shall be used to retrieve or reset sensor data.

Controller: (GET) Sensor Data Retrieval

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) SENSOR_VALUE	(PDL) 0x01
(PD) <div>Sensor # Requested</div>		

Response: (GET)

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND_RESPONSE	(PID) SENSOR_VALUE	(PDL) 0x09
(PD) <div><div>Sensor # Requested</div><div>Present Value (16-bit)</div><div>Lowest Detected Value (16-bit)</div><div>Highest Detected Value (16-bit)</div><div>Recorded Value (16-bit)</div></div>		

Controller: (SET) Sensor Data Reset

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) SENSOR_VALUE	(PDL) 0x01
(PD) <div>Sensor #</div>		

Response: (SET)

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND_RESPONSE	(PID) SENSOR_VALUE	(PDL) 0x09
(PD) <div><div>Sensor #</div><div>Present Value (16-bit)</div><div>Lowest Detected Value (16-bit)</div><div>Highest Detected Value (16-bit)</div><div>Recorded Value (16-bit)</div></div>		

The SET_COMMAND may be used in conjunction with SENSOR_VALUE to reset or clear a given sensor. When a sensor is successfully reset the values for Present, Lowest/Highest, Recorded values in the response message shall all be equal.

Data Description:

Sensor Number:

The sensor number is in the range 0x00 to 0xFE. A value 0xFF is used to represent all sensors for the Set command.

Present Value:

This is a 2's compliment signed 16-bit value that represents the present value of the sensor data.

Lowest Detected Value:

This is a 2's compliment signed 16-bit value that represents the lowest value registered by the sensor. Support for this data is optional.

Highest Detected Value:

This is a 2's compliment signed 16-bit value that represents the highest value registered by the sensor. Support for this data is optional.

Recorded Value:

This is a 2's compliment signed 16-bit value that represents the value that was recorded when the last RECORD_SENSORS was issued. Support for this data is optional.

10.7.3 Record Sensors (RECORD_SENSORS)

This parameter instructs devices such as dimming racks that monitor load changes to store the current value for monitoring sensor changes.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) SET_COMMAND	(PID) RECORD_SENSORS		(PDL) 0x01
(PD)			
<div>Sensor Number</div>			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) SET_COMMAND_RESPONSE	(PID) RECORD_SENSORS		(PDL) 0x00
(PD) Not Present			

The Sensor Number identifies the sensor that is recorded. A Sensor Number value of 0xFF indicates that all sensors shall be recorded.

Any values recorded in this manner shall be available for retrieval using the GET: SENSOR_VALUE message.

10.8 Power/Lamp Setting Parameter Messages

10.8.1 Get/Set Device Hours (DEVICE_HOURS)

This parameter is used to retrieve or set the number of hours of operation the device has been in use. Some devices may only support the GET_COMMAND for this operation and not allow the device's hours to be set.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) DEVICE_HOURS		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) DEVICE_HOURS		(PDL) 0x04
(PD)			
Device Hours (32-bit)			

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) SET_COMMAND	(PID) DEVICE_HOURS		(PDL) 0x04
(PD)			
Device Hours (32-bit)			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) SET_COMMAND_ RESPONSE	(PID) DEVICE_HOURS		(PDL) 0x00
(PD) Not Present			

10.8.2 Get/Set Lamp Hours (LAMP_HOURS)

This parameter is used to retrieve the number of lamp hours or to set the counter in the device to a specific starting value.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) LAMP_HOURS	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) LAMP_HOURS	(PDL) 0x04
(PD) Lamp Hours (32-bit)		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) LAMP_HOURS	(PDL) 0x04
(PD) Lamp Hours (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) LAMP_HOURS	(PDL) 0x00
(PD) Not Present		

The lamp hours are the total number of hours that the lamp has been on.

10.8.3 Get/Set Lamp Strikes (LAMP_STRIKES)

This parameter is used to retrieve the number of lamp strikes or to set the counter in the device to a specific starting value.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) LAMP_STRIKES	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) LAMP_STRIKES	(PDL) 0x04
(PD) Lamp Strikes (32-bit)		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) LAMP_STRIKES	(PDL) 0x04
(PD) Lamp Strikes (32-bit)		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) LAMP_STRIKES	(PDL) 0x00
(PD) Not Present		

The lamp strikes are incremented each time the lamp is struck or switched on.

10.8.4 Get/Set Lamp State (LAMP_STATE)

This parameter is used to retrieve or change the current operating state of the lamp.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) LAMP_STATE	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) LAMP_STATE	(PDL) 0x01
(PD) <div>Lamp State</div>		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) LAMP_STATE	(PDL) 0x01
(PD) <div>Lamp State</div>		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) LAMP_STATE	(PDL) 0x00
(PD) Not Present		

Lamp States are enumerated by Table A-8.

10.8.5 Get/Set Lamp On Mode (LAMP_ON_MODE)

This parameter is used to retrieve or change the current Lamp On Mode. Lamp On Mode defines the conditions under which a lamp will be struck.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) LAMP_ON_MODE	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) LAMP_ON_MODE	(PDL) 0x01
(PD) <div>Lamp On Mode</div>		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) LAMP_ON_MODE	(PDL) 0x01
(PD) <div>Lamp On Mode</div>		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) LAMP_ON_MODE	(PDL) 0x00
(PD) Not Present		

Lamp On Modes are enumerated by
Table A-9.

10.8.6 Get/Set Device Power Cycles (DEVICE_POWER_CYCLES)

This parameter is used to retrieve or set the number of Power-up cycles for the device. Some devices may only support the GET_COMMAND for this operation and not allow the device's power-up cycles to be set.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) DEVICE_POWER_CYCLES	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) DEVICE_POWER_CYCLES	(PDL) 0x04
(PD) <div>Power Cycle Count (32-bit)</div>		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) DEVICE_POWER_CYCLES	(PDL) 0x04
(PD) <div>Power Cycle Count (32-bit)</div>		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) DEVICE_POWER_CYCLES	(PDL) 0x00
(PD) Not Present		

10.9 Display Setting Parameter Messages

10.9.1 Get/Set Display Invert (DISPLAY_INVERT)

This parameter is used to retrieve or change the Display Invert setting. Invert is often used to rotate the display image by 180 degrees.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) DISPLAY_INVERT		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) DISPLAY_INVERT		(PDL) 0x01
(PD) <div>Off/On/Auto (0/1/2)</div>			

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 0xFFFF	
(CC) SET_COMMAND	(PID) DISPLAY_INVERT		(PDL) 0x01
(PD) <div>Off/On/Auto (0/1/2)</div>			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) SET_COMMAND_ RESPONSE	(PID) DISPLAY_INVERT		(PDL) 0x00
(PD) Not Present			

The 'Auto' setting is for devices that have the ability to automatically rotate the display based on the orientation of the device. Devices not supporting this functionality shall send a NACK Reason of NR_DATA_OUT_OF_RANGE.

10.9.2 Get/Set Display Level (DISPLAY_LEVEL)

This parameter is used to retrieve or change the Display Intensity setting.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) DISPLAY_LEVEL	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) DISPLAY_LEVEL	(PDL) 0x01
(PD) <div>Display Level</div>		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) DISPLAY_LEVEL	(PDL) 0x01
(PD) <div>Display Level</div>		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) DISPLAY_LEVEL	(PDL) 0x00
(PD) Not Present		

To turn the display off, Display Level shall be set to 0. To turn the display on full, Display Level shall be set to 0xFF. Any value in between represents a relative intensity setting. If the device does not support relative intensity settings, any non-zero value shall be interpreted as on.

The Display Level setting shall not override the use of the IDENTIFY_DEVICE Parameter for devices that use the display as part of the identification means.

10.10 Device Configuration Parameter Messages

10.10.1 Get/Set Pan Invert (PAN_INVERT)

This parameter is used to retrieve or change the Pan Invert setting.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) GET_COMMAND	(PID) PAN_INVERT		(PDL) 0x00
(PD) Not Present			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) PAN_INVERT		(PDL) 0x01
(PD) <div>Off/On (0/1)</div>			

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) SET_COMMAND	(PID) PAN_INVERT		(PDL) 0x01
<div>(PD)<div>Off/On (0/1)</div></div>			

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) SET_COMMAND_ RESPONSE	(PID) PAN_INVERT		(PDL) 0x00
(PD) Not Present			

10.10.2 Get/Set Tilt Invert (TILT_INVERT)

This parameter is used to retrieve or change the Tilt Invert setting.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) TILT_INVERT	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) TILT_INVERT	(PDL) 0x01
(PD) <div>Off/On (0/1)</div>		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) TILT_INVERT	(PDL) 0x01
(PD) <div>Off/On (0/1)</div>		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) TILT_INVERT	(PDL) 0x00
(PD) Not Present		

10.10.3 Get/Set Pan/Tilt Swap (PAN_TILT_SWAP)

This parameter is used to retrieve or change the Pan/Tilt Swap setting.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) PAN_TILT_SWAP	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) PAN_TILT_SWAP	(PDL) 0x01
(PD) <div>Off/On (0/1)</div>		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) PAN_TILT_SWAP	(PDL) 0x01
(PD) <div>Off/On (0/1)</div>		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) PAN_TILT_SWAP	(PDL) 0x00
(PD) Not Present		

10.10.4 Get/Set Device Real-Time Clock (REAL_TIME_CLOCK)

This parameter is used to retrieve or set the real-time clock in a device.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) REAL_TIME_CLOCK	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) GET_COMMAND_ RESPONSE	(PID) REAL_TIME_CLOCK		(PDL) 0x07
(PD)			
Year (16-bit)		Month	Day
Hour	Minute	Second	

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF	
(CC) SET_COMMAND	(PID) REAL_TIME_CLOCK		(PDL) 0x07
(PD)			
Year (16-bit)		Month	Day
Hour	Minute	Second	

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) REAL_TIME_CLOCK	(PDL) 0x00
(PD) Not Present		

Date and Time fields shall follow the format from Table 10-2. Hours shall be encoded using 24 hour format.

Table 10-2: Date and Time Ranges

	Minimum Value	Maximum Value
Year	2003	65535
Month	1	12
Day	1	31
Hour	0	23
Minute	0	59

10.11 Device Control Parameter Messages

10.11.1 Get/Set Identify Device (IDENTIFY_DEVICE)

This parameter is used for the user to physically identify the device represented by the UID.

The responder shall physically identify itself using a visible or audible action. For example, strobing a light or outputting fog.

The current state of a responders identification status may be obtained using a GET:IDENTIFY_DEVICE message.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) IDENTIFY_DEVICE	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) IDENTIFY_DEVICE	(PDL) 0x01
(PD)		
<div>Identify State Off/On (0/1)</div>		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) IDENTIFY_DEVICE	(PDL) 0x01
(PD) Identify Stop/Start (0/1)		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_RESPONSE	(PID) IDENTIFY_DEVICE	(PDL) 0x00
(PD) Not Present		

10.11.2 Reset Device (RESET_DEVICE)

This parameter is used to instruct the responder to reset itself. This parameter shall also clear the Discovery Mute flag. A cold reset is the equivalent of removing and reapplying power to the device.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) RESET_DEVICE	(PDL) 0x01
(PD) 0x01 for Warm Reset 0xFF for Cold Reset		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_RESPONSE	(PID) RESET_DEVICE	(PDL) 0x00
(PD) Not Present		

10.11.3 Get/Set Power State (POWER_STATE)

This parameter is used to retrieve or change the current device Power State. Power State specifies the current operating mode of the device.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) POWER_STATE	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) POWER_STATE	(PDL) 0x01
(PD) <div>Power State</div>		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) POWER_STATE	(PDL) 0x01
(PD) <div>Power State</div>		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) POWER_STATE	(PDL) 0x00
(PD) Not Present		

Power State Modes are enumerated by Table A-11.

10.11.4 Get/Set Perform Self Test (PERFORM_SELFTEST)

The PERFORM_SELFTEST message is used to execute any built in Self-Test routine that may be present. The test may run continuously until receiving a PERFORM_SELFTEST with a SELF_TEST_OFF value, or may exit upon completion.

The GET_COMMAND may be used to determine if any Self Tests are currently running. If any Self Tests are running the response flag shall be set to 0x01. A Self Test operation may return pass/fail status or other information by queuing a Status Message response.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) PERFORM_SELFTEST	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_RESPONSE	(PID) PERFORM_SELFTEST	(PDL) 0x01
(PD) <div>Self Tests Active TRUE/FALSE (1/0)</div>		

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) PERFORM_SELFTEST	(PDL) 0x01
(PD) <div>Self Test #</div>		

The Parameter Data includes a value indicating the self-test to execute as shown in Table A-10.

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_RESPONSE	(PID) PERFORM_SELFTEST	(PDL) 0x00
(PD) Not Present		

10.11.5 Get Self Test Description (SELF_TEST_DESCRIPTION)

This parameter is used to get a descriptive ASCII text label for a given Self Test Operation. The label may be up to 32 characters.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) SELF_TEST_DESCRIPTION	(PDL) 0x01
(PD) <div>Self Test # Requested</div>		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) SELF_TEST_DESCRIPTION	(PDL) 1-33 (1+Number of characters being sent)
(PD) <div>Self Test # Requested</div> <div>ASCII text label. Up to 32 characters.</div>		

10.11.6 Capture Preset (CAPTURE_PRESET)

This parameter is used to capture a static scene into a Preset within the responder. The actual data that is captured is beyond the scope of this standard. Upon receipt of this parameter the responder shall capture the scene and store it to the designated preset.

Fade and Wait times for building sequences may also be included. Times are in tenths of a second. When timing information is not required the fields shall be set to 0x00.

The Up Fade Time is the fade in time for the current scene and the Down Fade Time is the down fade for the previous scene or active look. The Wait Time is the time the device spends holding the current scene before proceeding to play the next scene when the presets are being played back as a sequence using PRESET_PLAYBACK_ALL.

Controller:

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF					
(CC) SET_COMMAND	(PID) CAPTURE_PRESET		(PDL) 0x02 or 0x08				
(PD)							
<table><tr><td>Scene # (16-bit)</td></tr><tr><td>Up Fade Time (16-bit)</td></tr><tr><td>Down Fade Time (16-bit)</td></tr><tr><td>Wait Time (16-bit)</td></tr></table>				Scene # (16-bit)	Up Fade Time (16-bit)	Down Fade Time (16-bit)	Wait Time (16-bit)
Scene # (16-bit)							
Up Fade Time (16-bit)							
Down Fade Time (16-bit)							
Wait Time (16-bit)							

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD	
(CC) SET_COMMAND_RESPONSE	(PID) CAPTURE_PRESET		(PDL) 0x00
(PD) Not Present			

10.11.7 Get/Set Preset Playback (PRESET_PLAYBACK)

This parameter is used to recall pre-recorded Presets.

Preset playback types are enumerated by Table A-7.

Controller: (GET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) GET_COMMAND	(PID) PRESET_PLAYBACK	(PDL) 0x00
(PD) Not Present		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) GET_COMMAND_ RESPONSE	(PID) PRESET_PLAYBACK	(PDL) 0x03
(PD) Mode (16-bit) (OFF/ALL/Scene #) Level (0x00-0xFF)		

Controller: (SET)

(Port ID) 0x00 - 0xFF	(Message Count) 0x00	(Sub-Device) 0x0000 (Root) or 0x0001- 0x0200 or 0xFFFF
(CC) SET_COMMAND	(PID) PRESET_PLAYBACK	(PDL) 0x03
(PD) Mode (16-bit) (OFF/ALL/Scene #) Level (0x00-0xFF)		

Response:

(Response Type) ACK	(Message Count) 0x00-0xFF	(Sub-Device) Copy of Controller SD
(CC) SET_COMMAND_ RESPONSE	(PID) PRESET_PLAYBACK	(PDL) 0x00
(PD) Not Present		

The Level field allows a Master Fader value to be applied to scale the output of the Preset Playback. A Level value of 0x00 means preset scaled at 0 and a level value of 0xFF represents a Preset scaled at full.

If the controller is not capable of supporting the Level capabilities, then it shall set the value to 0xFF.

Setting the Preset Playback Mode to OFF shall restore the device to respond to incoming Null Start Code DMX512 packets, while setting the Level to 0 would leave Preset Playback mode active, but with a zero output intensity level.

Setting Preset Playback Mode to ALL shall play the scenes together in a looped sequence for devices supporting that operation.

Setting the Preset Playback Mode to an individual Scene number shall play an individual scene.

11 Operational Issues

11.1 Polling Intervals

Responders should be polled at a regular interval once discovery is complete. This may be done using QUEUED_MESSAGE, STATUS_MESSAGES, or other PID's as appropriate to the application.

The use of QUEUED_MESSAGE as the primary polling message allows the most efficient return of queued messages and status information. The use of STATUS_MESSAGES as the primary polling message allows increased control over the amount of status information returned, while still providing an indication of the number of Queued Messages pending.

Any response, including a NACK, to any message indicates the device is still present on the network.

Responders should not be polled more often than once every five seconds unless in a diagnostics mode or the user is performing an action that requires direct communication.

12 Protocol Support Issues

The message structure defined in this document is intended to allow levels of implementation ranging from the most basic to a full message implementation. The Required column Table A-3 defines the minimum set of PIDs that a responder shall support.

Appendix A: Defined Parameters (Normative)

START Codes (Slot 0)

SC_RDM 0xCC

RDM Protocol Data Structure ID's (Slot 1)

SC_SUB_MESSAGE 0x01

Broadcast Device UID's

BROADCAST_ALL_DEVICES_ID (Broadcast all Manufacturers) 0xFFFFFFFFFFFF
ALL_DEVICES_ID (Specific Manufacturer ID 0xmmm) 0xmmmFFFFFFFF

SUB_DEVICE_ALL_CALL 0xFFFF

Table A-1: Command Class Defines

RDM Command Classes (Slot 20)	Value	Comment
DISCOVERY_COMMAND	0x10	
DISCOVERY_COMMAND_RESPONSE	0x11	
GET_COMMAND	0x20	
GET_COMMAND_RESPONSE	0x21	
SET_COMMAND	0x30	
SET_COMMAND_RESPONSE	0x31	

Table A-2: Response Type Defines

RDM Response Type (Slot 16)	Value	Comment
RESPONSE_TYPE_ACK	0x00	
RESPONSE_TYPE_ACK_TIMER	0x01	
RESPONSE_TYPE_NACK_REASON	0x02	See Table A-17
RESPONSE_TYPE_ACK_OVERFLOW	0x03	Additional Response Data available beyond single response length.

Table A-3: RDM Categories/Parameter ID Defines

GET Allowed	SET Allowed	RDM Parameter ID's (Slot 21-22)	Value	Comment	Required
		Category – Network Management			
		DISC_UNIQUE_BRANCH	0x0001		✓
		DISC_MUTE	0x0002		✓
		DISC_UN_MUTE	0x0003		✓
✓		PROXIED_DEVICES	0x0010		
✓		PROXIED_DEVICE_COUNT	0x0011		
✓	✓	COMMS_STATUS	0x0015		
		Category - Status Collection			
✓		QUEUED_MESSAGE	0x0020	See Table A-4	
✓		STATUS_MESSAGES	0x0030	See Table A-4	
✓		STATUS_ID_DESCRIPTION	0x0031		
	✓	CLEAR_STATUS_ID	0x0032		
✓	✓	SUB_DEVICE_STATUS_REPORT_THR ESHOLD	0x0033	See Table A-4	
		Category - RDM Information			
✓		SUPPORTED_PARAMETERS	0x0050	* Support required only if supporting Parameters beyond the minimum required set.	✓*
✓		PARAMETER_DESCRIPTION	0x0051	- Support required for Manufacturer-Specific PIDs exposed in SUPPORTED_PARAMETERS message.	✓-
		Category – Product Information			
✓		DEVICE_INFO	0x0060		✓
✓		PRODUCT_DETAIL_ID_LIST	0x0070		
✓		DEVICE_MODEL_DESCRIPTION	0x0080		
✓		MANUFACTURER_LABEL	0x0081		
✓	✓	DEVICE_LABEL	0x0082		
✓	✓	FACTORY_DEFAULTS	0x0090		
✓		LANGUAGE_CAPABILITIES	0x00A0		
✓	✓	LANGUAGE	0x00B0		
✓		SOFTWARE_VERSION_LABEL	0x00C0		✓
✓		BOOT_SOFTWARE_VERSION_ID	0x00C1		
✓		BOOT_SOFTWARE_VERSION_LABEL	0x00C2		
		Category - DMX512 Setup			
✓	✓	DMX_PERSONALITY	0x00E0		
✓		DMX_PERSONALITY_DESCRIPTION	0x00E1		
✓	✓	DMX_START_ADDRESS	0x00F0	* Support required if device uses a DMX512 Slot.	✓*
✓		SLOT_INFO	0x0120		
✓		SLOT_DESCRIPTION	0x0121		
✓		DEFAULT_SLOT_VALUE	0x0122		
		Category – Sensors	0x02xx		
✓		SENSOR_DEFINITION	0x0200		
✓	✓	SENSOR_VALUE	0x0201		

GET Allowed	SET Allowed	RDM Parameter ID's (Slot 21-22)	Value	Comment	Required
	✓	RECORD_SENSORS	0x0202		
		Category – Dimmer Settings	0x03xx	Future	
		Category – Power/Lamp Settings	0x04xx		
✓	✓	DEVICE_HOURS	0x0400		
✓	✓	LAMP_HOURS	0x0401		
✓	✓	LAMP_STRIKES	0x0402		
✓	✓	LAMP_STATE	0x0403	See Table A-8	
✓	✓	LAMP_ON_MODE	0x0404	See Table A-9	
✓	✓	DEVICE_POWER_CYCLES	0x0405		
		Category - Display Settings	0x05xx		
✓	✓	DISPLAY_INVERT	0x0500		
✓	✓	DISPLAY_LEVEL	0x0501		
		Category – Configuration	0x06xx		
✓	✓	PAN_INVERT	0x0600		
✓	✓	TILT_INVERT	0x0601		
✓	✓	PAN_TILT_SWAP	0x0602		
✓	✓	REAL_TIME_CLOCK	0x0603		
		Category – Control	0x10xx		
✓	✓	IDENTIFY_DEVICE	0x1000		✓
	✓	RESET_DEVICE	0x1001		
✓	✓	POWER_STATE	0x1010	See Table A-11	
✓	✓	PERFORM_SELFTEST	0x1020	See Table A-10	
✓		SELF_TEST_DESCRIPTION	0x1021		
	✓	CAPTURE_PRESET	0x1030		
✓	✓	PRESET_PLAYBACK	0x1031	See Table A-7	
		ESTA Reserved Future RDM Development	0x7FE0-0x7FFF		
		Manufacturer-Specific PIDs	0x8000-0xFFDF		
		ESTA Reserved Future RDM Development	0xFFE0-0xFFFF		

Table A-4: Status Type Defines

Status Type Defines	Value	Comment
STATUS_NONE	0x00	Not allowed for use with GET: QUEUED_MESSAGE
STATUS_GET_LAST_MESSAGE	0x01	
STATUS_ADVISORY	0x02	
STATUS_WARNING	0x03	
STATUS_ERROR	0x04	

Table A-5: Product Category Defines

Product Category Defines	MSB Coarse	LSB Fine	Comment
PRODUCT_CATEGORY_NOT_DECLARED	0x00	0x00	
Fixtures – intended as source of illumination			See Note 1
PRODUCT_CATEGORY_FIXTURE	0x01	0x00	No Fine Category declared
PRODUCT_CATEGORY_FIXTURE_FIXED	0x01	0x01	No pan / tilt / focus style functions
PRODUCT_CATEGORY_FIXTURE_MOVING_YOKE	0x01	0x02	
PRODUCT_CATEGORY_FIXTURE_MOVING_MIRROR	0x01	0x03	
PRODUCT_CATEGORY_FIXTURE_OTHER	0x01	0xFF	For example, focus but no pan/tilt.
Fixture Accessories – add-ons to fixtures or projectors			
PRODUCT_CATEGORY_FIXTURE_ACCESSORY	0x02	0x00	No Fine Category declared.
PRODUCT_CATEGORY_FIXTURE_ACCESSORY_COLOR	0x02	0x01	Scrollers / Color Changers
PRODUCT_CATEGORY_FIXTURE_ACCESSORY_YOKE	0x02	0x02	Yoke add-on
PRODUCT_CATEGORY_FIXTURE_ACCESSORY_MIRROR	0x02	0x03	Moving mirror add-on
PRODUCT_CATEGORY_FIXTURE_ACCESSORY_EFFECT	0x02	0x04	Effects Discs
PRODUCT_CATEGORY_FIXTURE_ACCESSORY_BEAM	0x02	0x05	Gobo Rotators / Iris / Shutters / Dousers Beam modifiers.
PRODUCT_CATEGORY_FIXTURE_ACCESSORY_OTHER	0x02	0xFF	
Projectors - light source capable of producing realistic images from another media			Video / Slide / Oil Wheel / Film / LCD
PRODUCT_CATEGORY_PROJECTOR	0x03	0x00	No Fine Category declared.
PRODUCT_CATEGORY_PROJECTOR_FIXED	0x03	0x01	No pan / tilt functions.
PRODUCT_CATEGORY_PROJECTOR_MOVING_YOKE	0x03	0x02	
PRODUCT_CATEGORY_PROJECTOR_MOVING_MIRROR	0x03	0x03	
PRODUCT_CATEGORY_PROJECTOR_OTHER	0x03	0xFF	
Atmospheric Effect – earth/wind/fire			
PRODUCT_CATEGORY_ATMOSPHERIC	0x04	0x00	No Fine Category declared.
PRODUCT_CATEGORY_ATMOSPHERIC_EFFECT	0x04	0x01	Fogger / Hazer / Flame, etc.
PRODUCT_CATEGORY_ATMOSPHERIC_PYRO	0x04	0x02	See Note 2.
PRODUCT_CATEGORY_ATMOSPHERIC_OTHER	0x04	0xFF	
Intensity Control (specifically Dimming equipment)			
PRODUCT_CATEGORY_DIMMER	0x05	0x00	No Fine Category declared.

PRODUCT_CATEGORY_DIMMER_AC_INCANDESCENT	0x05	0x01	AC > 50VAC
PRODUCT_CATEGORY_DIMMER_AC_FLUORESCENT	0x05	0x02	
PRODUCT_CATEGORY_DIMMER_AC_COLD CATHODE	0x05	0x03	High Voltage outputs such as Neon or other cold cathode.
PRODUCT_CATEGORY_DIMMER_AC_NONDIM	0x05	0x04	Non-Dim module in dimmer rack.
PRODUCT_CATEGORY_DIMMER_AC_ELV	0x05	0x05	AC <= 50V such as 12/24V AC Low voltage lamps.
PRODUCT_CATEGORY_DIMMER_AC_OTHER	0x05	0x06	
PRODUCT_CATEGORY_DIMMER_DC_LEVEL	0x05	0x07	Variable DC level output.
PRODUCT_CATEGORY_DIMMER_DC_PWM	0x05	0x08	Chopped (PWM) output.
PRODUCT_CATEGORY_DIMMER_CS_LED	0x05	0x09	Specialized LED dimmer.
PRODUCT_CATEGORY_DIMMER_OTHER	0x05	0xFF	
Power Control (other than Dimming equipment)			
PRODUCT_CATEGORY_POWER	0x06	0x00	No Fine Category declared.
PRODUCT_CATEGORY_POWER_CONTROL	0x06	0x01	Contactors racks, other forms of Power Controllers.
PRODUCT_CATEGORY_POWER_SOURCE	0x06	0x02	Generators
PRODUCT_CATEGORY_POWER_OTHER	0x06	0xFF	
Scenic Drive – including motorized effects unrelated to light source.			
PRODUCT_CATEGORY_SCENIC	0x07	0x00	No Fine Category declared
PRODUCT_CATEGORY_SCENIC_DRIVE	0x07	0x01	Rotators / Kabuki drops, etc. See Note 2.
PRODUCT_CATEGORY_SCENIC_OTHER	0x07	0xFF	
DMX Infrastructure, conversion and interfaces			
PRODUCT_CATEGORY_DATA	0x08	0x00	No Fine Category declared.
PRODUCT_CATEGORY_DATA_DISTRIBUTION	0x08	0x01	Splitters / repeaters / data patch / Ethernet products used to distribute DMX universes.
PRODUCT_CATEGORY_DATA_CONVERSION	0x08	0x02	Protocol Conversion analog decoders.
PRODUCT_CATEGORY_DATA_OTHER	0x08	0xFF	
Audio-Visual Equipment			
PRODUCT_CATEGORY_AV	0x09	0x00	No Fine Category declared.
PRODUCT_CATEGORY_AV_AUDIO	0x09	0x01	Audio controller or device.
PRODUCT_CATEGORY_AV_VIDEO	0x09	0x02	Video controller or display device.
PRODUCT_CATEGORY_AV_OTHER	0x09	0xFF	
Parameter Monitoring Equipment			
			See Note 3.

PRODUCT_CATEGORY_MONITOR	0x0A	0x00	No Fine Category declared.
PRODUCT_CATEGORY_MONITOR_ACLINEPOWER	0x0A	0x01	Product that monitors AC line voltage, current or power.
PRODUCT_CATEGORY_MONITOR_DCPOWER	0x0A	0x02	Product that monitors DC line voltage, current or power.
PRODUCT_CATEGORY_MONITOR_ENVIRONMENTAL	0x0A	0x03	Temperature or other environmental parameter.
PRODUCT_CATEGORY_MONITOR_OTHER	0x0A	0xFF	
Controllers, Backup devices			
PRODUCT_CATEGORY_CONTROL	0x70	0x00	No Fine Category declared.
PRODUCT_CATEGORY_CONTROL_CONTROLLER	0x70	0x01	
PRODUCT_CATEGORY_CONTROL_BACKUPDEVICE	0x70	0x02	
PRODUCT_CATEGORY_CONTROL_OTHER	0x70	0xFF	
Test Equipment			
PRODUCT_CATEGORY_TEST	0x71	0x00	No Fine Category declared.
PRODUCT_CATEGORY_TEST_EQUIPMENT	0x71	0x01	
PRODUCT_CATEGORY_TEST_EQUIPMENT_OTHER	0x71	0xFF	
Miscellaneous			
PRODUCT_CATEGORY_OTHER	0x7F	0xFF	For devices that aren't described within this table.
Manufacturer Specific Categories	0x80 - 0xDF	0x00 - 0xFF	

Note 1: A fixture in this context is defined as a light source intended as a means of illumination. A projector is a light source capable of producing realistic images from another media. A light source intended for use with Fiber Optics should be classified as a fixture.

Note 2: The DMX512 standard specifically states that, as there is no mandatory error checking, DMX512 is not an appropriate control protocol for hazardous applications. RDM may, however, be appropriate for configuration and monitoring purposes.

Note 3: This category is for equipment that has no DMX512 control capability, but uses RDM to provide a data logging or monitoring function.

Table A-6: Product Detail Defines

Product Detail ID Defines	Value	Comment
PRODUCT_DETAIL_NOT DECLARED	0x0000	
Generally applied to fixtures		
PRODUCT_DETAIL_ARC	0x0001	Intended for constant light output.

PRODUCT_DETAIL_METAL_HALIDE	0x0002	
PRODUCT_DETAIL_INCANDESCENT	0x0003	
PRODUCT_DETAIL_LED	0x0004	
PRODUCT_DETAIL_FLUORESCENT	0x0005	
PRODUCT_DETAIL_COLD CATHODE	0x0006	includes Neon/Argon
PRODUCT_DETAIL_ELECTROLUMINESCENT	0x0007	
PRODUCT_DETAIL_LASER	0x0008	
PRODUCT_DETAIL_FLASHTUBE	0x0009	Strobes or other flashtubes
Generally applied to fixture accessories		
PRODUCT_DETAIL_COLORSCROLLER	0x0100	
PRODUCT_DETAIL_COLORWHEEL	0x0101	
PRODUCT_DETAIL_COLORCHANGE	0x0102	Semaphore or other type
PRODUCT_DETAIL_IRIS_DOUSER	0x0103	
PRODUCT_DETAIL_DIMMING_SHUTTER	0x0104	
PRODUCT_DETAIL_PROFILE_SHUTTER	0x0105	hard-edge beam masking
PRODUCT_DETAIL_BARNDOOR_SHUTTER	0x0106	soft-edge beam masking
PRODUCT_DETAIL_EFFECTS_DISC	0x0107	
PRODUCT_DETAIL_GOBO_ROTATOR	0x0108	
Generally applied to Projectors		
PRODUCT_DETAIL_VIDEO	0x0200	
PRODUCT_DETAIL_SLIDE	0x0201	
PRODUCT_DETAIL_FILM	0x0202	
PRODUCT_DETAIL_OILWHEEL	0x0203	
PRODUCT_DETAIL_LCDGATE	0x0204	
Generally applied to Atmospheric Effects		
PRODUCT_DETAIL_FOGGER_GLYCOL	0x0300	Glycol/Glycerin hazer
PRODUCT_DETAIL_FOGGER_MINERAL OIL	0x0301	White Mineral oil hazer
PRODUCT_DETAIL_FOGGER_WATER	0x0302	Water hazer
PRODUCT_DETAIL_CO2	0x0303	Dry Ice/Carbon Dioxide based
PRODUCT_DETAIL_LN2	0x0304	Nitrogen based
PRODUCT_DETAIL_BUBBLE	0x0305	including foam
PRODUCT_DETAIL_FLAME_PROpane	0x0306	
PRODUCT_DETAIL_FLAME_OTHER	0x0307	

PRODUCT_DETAIL_OLEFACTORY_STIMULATOR	0x0308	Scents
PRODUCT_DETAIL_SNOW	0x0309	
PRODUCT_DETAIL_WATER_JET	0x030A	Fountain controls etc
PRODUCT_DETAIL_WIND	0x030B	Air Mover
PRODUCT_DETAIL_CONFETTI	0x030C	
PRODUCT_DETAIL_HAZARD	0x030D	Any form of pyrotechnic control or device.
Generally applied to Dimmers/Power controllers		See Note 1
PRODUCT_DETAIL_PHASE_CONTROL	0x0400	
PRODUCT_DETAIL_REVERSE_PHASE_CONTROL	0x0401	includes FET/IGBT
PRODUCT_DETAIL_SINE	0x0402	
PRODUCT_DETAIL_PWM	0x0403	
PRODUCT_DETAIL_DC	0x0404	Variable voltage
PRODUCT_DETAIL_HFBALLAST	0x0405	for Fluorescent
PRODUCT_DETAIL_HFHV_NEONBALLAST	0x0406	for Neon/Argon and other coldcathode.
PRODUCT_DETAIL_HFHV_EL	0x0407	for Electroluminescent
PRODUCT_DETAIL_MHR_BALLAST	0x0408	for Metal Halide
PRODUCT_DETAIL_BITANGLE_MODULATION	0x0409	
PRODUCT_DETAIL_FREQUENCY_MODULATION	0x040A	
PRODUCT_DETAIL_HIGHFREQUENCY_12V	0x040B	as commonly used with MR16 lamps
PRODUCT_DETAIL_RELAY_MECHANICAL	0x040C	See Note 1
PRODUCT_DETAIL_RELAY_ELECTRONIC	0x040D	See Note 1, Note 2
PRODUCT_DETAIL_SWITCH_ELECTRONIC	0x040E	See Note 1, Note 2
PRODUCT_DETAIL_CONTACTOR	0x040F	See Note 1
Generally applied to Scenic drive		
PRODUCT_DETAIL_MIRRORBALL_ROTATOR	0x0500	
PRODUCT_DETAIL_OTHER_ROTATOR	0x0501	includes turntables
PRODUCT_DETAIL_KABUKI_DROP	0x0502	
PRODUCT_DETAIL_CURTAIN	0x0503	flown or traveller
PRODUCT_DETAIL_LINESET	0x0504	
PRODUCT_DETAIL_MOTOR_CONTROL	0x0505	
PRODUCT_DETAIL_DAMPER_CONTROL	0x0506	HVAC Damper
Generally applied to Data Distribution		

PRODUCT_DETAIL_SPLITTER	0x0600	Includes buffers/repeaters
PRODUCT_DETAIL_ETHERNET_NODE	0x0601	DMX512 to/from Ethernet
PRODUCT_DETAIL_MERGE	0x0602	DMX512 combiner
PRODUCT_DETAIL_DATAPATCH	0x0603	Electronic Datalink Patch
PRODUCT_DETAIL_WIRELESS_LINK	0x0604	radio/infrared
Generally applied to Data Conversion and Interfaces		
PRODUCT_DETAIL_PROTOCOL_CONVERTOR	0x0701	D54/AMX192/Non DMX serial links, etc to/from DMX512
PRODUCT_DETAIL_ANALOG_DEMULTIPLEX	0x0702	DMX to DC voltage
PRODUCT_DETAIL_ANALOG_MULTIPLEX	0x0703	DC Voltage to DMX
PRODUCT_DETAIL_SWITCH_PANEL	0x0704	Pushbuttons to DMX or polled using RDM
Generally applied to Audio or Video (AV) devices		
PRODUCT_DETAIL_ROUTER	0x0800	Switching device
PRODUCT_DETAIL_FADER	0x0801	Single channel
PRODUCT_DETAIL_MIXER	0x0802	Multi-channel
Generally applied to Controllers, Backup devices and Test Equipment		
PRODUCT_DETAIL_CHANGEOVER_MANUAL	0x0900	requires manual intervention to assume control of DMX line
PRODUCT_DETAIL_CHANGEOVER_AUTO	0x0901	may automatically assume control of DMX line
PRODUCT_DETAIL_TEST	0x0902	test equipment
Could be applied to any category		
PRODUCT_DETAIL_GFI_RCD	0x0A00	device includes GFI/RCD trip
PRODUCT_DETAIL_BATTERY	0x0A01	device is battery operated
PRODUCT_DETAIL_CONTROLLABLE_BREAKER	0x0A02	
Manufacturer Specific Types	0x8000-0xDFFF	
PRODUCT_DETAIL_OTHER	0x7FFF	for use where the Manufacturer believes that none of the defined details apply.

Note 1: Products intended for switching 50V AC / 120V DC or greater should be declared with a Product Category of PRODUCT_CATEGORY_POWER_CONTROL.

Products only suitable for extra low voltage switching (typically up to 50VAC / 30VDC) at currents less than 1 ampere should be declared with a Product Category of PRODUCT_CATEGORY_DATA_CONVERSION.

Please refer to GET: DEVICE_INFO and Table A-5 for an explanation of Product Category declaration.

Note 2: Products with TTL, MOSFET or Open Collector Transistor Outputs or similar non-isolated electronic outputs should be declared as PRODUCT_DETAIL_SWITCH_ELECTRONIC. Use of PRODUCT_DETAIL_RELAY_ELECTRONIC shall be restricted to devices whereby the switched circuits are electrically isolated from the control signals.

Table A-7: Preset Playback Defines

Preset Playback Defines	Value	Comment
PRESET_PLAYBACK_OFF	0x0000	Returns to Normal DMX512 Input
PRESET_PLAYBACK_ALL	0xFFFF	Plays Scenes in Sequence if supported.
PRESET_PLAYBACK_SCENE	0x0001-0xFFFE	Plays individual Scene #

Table A-8: Lamp State Defines

Lamp State Defines	Value	Comment
LAMP_OFF	0x00	No demonstrable light output
LAMP_ON	0x01	
LAMP_STRIKE	0x02	Arc-Lamp ignite
LAMP_STANDBY	0x03	Arc-Lamp Reduced Power Mode
LAMP_NOT_PRESENT	0x04	Lamp not installed
LAMP_ERROR	0x7F	
Manufacturer-Specific States	0x80 – 0xDF	

Table A-9: Lamp On Mode Defines

Lamp Mode Defines	Value	Comment
LAMP_ON_MODE_OFF	0x00	Lamp Stays off until directly instructed to Strike.
LAMP_ON_MODE_DMX	0x01	Lamp Strikes upon receiving a DMX512 signal.
LAMP_ON_MODE_ON	0x02	Lamp Strikes automatically at Power-up.
LAMP_ON_MODE_AFTER_CAL	0x03	Lamp Strikes after Calibration or Homing procedure.
Manufacturer-Specific Modes	0x80 – 0xDF	

Table A-10: Self Test Defines

Self Test Defines	Value	Comment
SELF_TEST_OFF	0x00	Turns Self Tests Off
Manufacturer Tests	0x01 – 0xFE	Various Manufacturer Self Tests
SELF_TEST_ALL	0xFF	Self Test All, if applicable

Table A-11: Power State Defines

Power State Defines	Value	Comment
POWER_STATE_FULL_OFF	0x00	Completely disengages power to device. Device can no longer respond.
POWER_STATE_SHUTDOWN	0x01	Reduced power mode, may require device reset to return to normal operation. Device still responds to messages.
POWER_STATE_STANDBY	0x02	Reduced power mode. Device can return to NORMAL without a reset. Device still responds to messages.
POWER_STATE_NORMAL	0xFF	Normal Operating Mode.

Table A-12: Sensor Type Defines

Sensor Type Defines	Value	Comment
SENS_TEMPERATURE	0x00	
SENS_VOLTAGE	0x01	
SENS_CURRENT	0x02	
SENS_FREQUENCY	0x03	
SENS_RESISTANCE	0x04	Eg: Cable resistance
SENS_POWER	0x05	
SENS_MASS	0x06	Eg: Truss load Cell
SENS_LENGTH	0x07	
SENS_AREA	0x08	
SENS_VOLUME	0x09	Eg: Smoke Fluid
SENS_DENSITY	0x0A	
SENS_VELOCITY	0x0B	
SENS_ACCELERATION	0x0C	
SENS_FORCE	0x0D	
SENS_ENERGY	0x0E	
SENS_PRESSURE	0x0F	
SENS_TIME	0x10	
SENS_ANGLE	0x11	
SENS_POSITION X	0x12	E.g.: Lamp position on Truss

SENS_POSITION Y	0x13	
SENS_POSITION Z	0x14	
SENS_ANGULAR_VELOCITY	0x15	E.g.: Wind speed
SENS_LUMINOUS_INTENSITY	0x16	
SENS_LUMINOUS_FLUX	0x17	
SENS_ILLUMINANCE	0x18	
SENS_CHROMINANCE_RED	0x19	
SENS_CHROMINANCE_GREEN	0x1A	
SENS_CHROMINANCE_BLUE	0x1B	
SENS_CONTACTS	0x1C	E.g.: Switch inputs.
SENS_MEMORY	0x1D	E.g.: ROM Size
SENS_ITEMS	0x1E	E.g.: Scroller gel frames.
SENS_HUMIDITY	0x1F	
SENS_COUNTER_16BIT	0x20	
SENS_OTHER	0x7F	
Manufacturer-Specific Sensors	0x80 – 0xFF	

Table A-13: Sensor Unit Defines

Sensor Unit Defines	Value	Table A-12 Reference
UNITS_NONE	0x00	CONTACTS
UNITS_CENTIGRADE	0x01	TEMPERATURE
UNITS_VOLTS_DC	0x02	VOLTAGE
UNITS_VOLTS_AC_PEAK	0x03	VOLTAGE
UNITS_VOLTS_AC_RMS	0x04	VOLTAGE
UNITS_AMPERE_DC	0x05	CURRENT
UNITS_AMPERE_AC_PEAK	0x06	CURRENT
UNITS_AMPERE_AC_RMS	0x07	CURRENT
UNITS_HERTZ	0x08	FREQUENCY / ANG_VEL
UNITS_OHM	0x09	RESISTANCE
UNITS_WATT	0x0A	POWER
UNITS_KILOGRAM	0x0B	MASS

Sensor Unit Defines	Value	Table A-12 Reference
UNITS_METERS	0x0C	LENGTH / POSITION
UNITS_METERS_SQUARED	0x0D	AREA
UNITS_METERS_CUBED	0x0E	VOLUME
UNITS_KILOGRAMMES_PER_METER_CUBED	0x0F	DENSITY
UNITS_METERS_PER_SECOND	0x10	VELOCITY
UNITS_METERS_PER_SECOND_SQUARED	0x11	ACCELERATION
UNITS_NEWTON	0x12	FORCE
UNITS_JOULE	0x13	ENERGY
UNITS_PASCAL	0x14	PRESSURE
UNITS_SECOND	0x15	TIME
UNITS_DEGREE	0x16	ANGLE
UNITS_STERADIAN	0x17	ANGLE
UNITS_CANDELA	0x18	LUMINOUS_INTENSITY
UNITS_LUMEN	0x19	LUMINOUS_FLUX
UNITS_LUX	0x1A	ILLUMINANCE
UNITS_IRE	0x1B	CHROMINANCE
UNITS_BYTE	0x1C	MEMORY
Manufacturer-Specific Units	0x80 – 0xFF	

Table A-14: Sensor Unit Prefix Defines

Sensor Unit Prefix Defines	Value	Description
PREFIX_NONE	0x00	Multiply by 1
PREFIX_DECI	0x01	Multiply by 10^{-1}
PREFIX_CENTI	0x02	Multiply by 10^{-2}
PREFIX_MILLI	0x03	Multiply by 10^{-3}
PREFIX_MICRO	0x04	Multiply by 10^{-6}
PREFIX_NANO	0x05	Multiply by 10^{-9}
PREFIX_PICO	0x06	Multiply by 10^{-12}
PREFIX_FEMPTO	0x07	Multiply by 10^{-15}
PREFIX_ATTO	0x08	Multiply by 10^{-18}
PREFIX_ZEPTO	0x09	Multiply by 10^{-21}
PREFIX_YOCTO	0x0A	Multiply by 10^{-24}
PREFIX_DECA	0x11	Multiply by 10^{+1}
PREFIX_HECTO	0x12	Multiply by 10^{+2}
PREFIX_KILO	0x13	Multiply by 10^{+3}
PREFIX_MEGA	0x14	Multiply by 10^{+6}
PREFIX_GIGA	0x15	Multiply by 10^{+9}
PREFIX_TERRA	0x16	Multiply by 10^{+12}
PREFIX_PETA	0x17	Multiply by 10^{+15}
PREFIX_EXA	0x18	Multiply by 10^{+18}
PREFIX_ZETTA	0x19	Multiply by 10^{+21}
PREFIX_YOTTA	0x1A	Multiply by 10^{+24}

Notes:

When a prefix is used with MEMORY, the multiplier refers to binary multiple. i.e. KILO means multiply by 1024.

When a prefix is used with MASS, note that the UNIT is Kilogram. The prefix PREFIX_MILLI is used to denote grams.

Table A-15: Data Type Defines

Data Type Defines	Value	Description
DS_NOT_DEFINED	0x00	Data type is not defined
DS_BIT_FIELD	0x01	Data is bit packed
DS_ASCII	0x02	Data is a string
DS_UNSIGNED_BYTE	0x03	Data is an array of unsigned bytes
DS_SIGNED_BYTE	0x04	Data is an array of signed bytes
DS_UNSIGNED_WORD	0x05	Data is an array of unsigned 16-bit words
DS_SIGNED_WORD	0x06	Data is an array of signed 16-bit words
DS_UNSIGNED_DWORD	0x07	Data is an array of unsigned 32-bit words
DS_SIGNED_DWORD	0x08	Data is an array of signed 32-bit words
Manufacturer-Specific Data Types	0x80 – 0xDF	

Table A-16 Parameter Description Command Class Defines

Command Class Defines	Value	Description
CC_GET	0x01	PID supports GET only
CC_SET	0x02	PID supports SET only
CC_GET_SET	0x03	PID supports GET & SET

Table A-17: Response NACK Reason Code Defines

Response NACK Reason Codes	Value	Description
NR_UNKNOWN_PID	0x0000	The responder cannot comply with request because the message is not implemented in responder.
NR_FORMAT_ERROR	0x0001	The responder cannot interpret request as controller data was not formatted correctly.
NR_HARDWARE_FAULT	0x0002	The responder cannot comply due to an internal hardware fault.
NR_PROXY_REJECT	0x0003	Proxy is not the RDM line master and cannot comply with message.
NR_WRITE_PROTECT	0x0004	SET Command normally allowed but being blocked currently.
NR_UNSUPPORTED_COMMAND_CLASS	0x0005	Not valid for Command Class attempted. May be used where GET allowed but SET is not supported.
NR_DATA_OUT_OF_RANGE	0x0006	Value for given Parameter out of allowable range or not supported.
NR_BUFFER_FULL	0x0007	Buffer or Queue space currently has no free space to store data.
NR_PACKET_SIZE_UNSUPPORTED	0x0008	Incoming message exceeds buffer capacity.
NR_SUB_DEVICE_OUT_OF_RANGE	0x0009	Sub-Device is out of range or unknown.

Appendix B: Status Message ID's (Normative)

Manufacturers shall use publicly defined Status Message ID codes rather than developing their own Manufacturer-specific codes whenever possible.

Publicly defined codes are in the range of 0x0000 - 0x7FFF. Manufacturer-specific codes are in the range of 0x8000 – 0xFFDF.

Informative note: The ESTA website (<http://www.esta.org/rdm>) may document new publicly defined Status Message ID codes in addition to those enumerated in this document. Implementors of this standard are encouraged to check the ESTA website for applicable Status Message ID's before creating manufacturer-specific codes.

Table B-2 shows some of the predefined Status Message ID definitions, including the value, what the numeric value represents, and the recommended Status ID Description associated with the ID.

The Status ID Descriptions may include a marker that indicates how the controller should interpret the optional parameter value, and where the value should be displayed in the message displayed to the user. The marker types are defined in Table B-1.

Table B-1: Status Message Markers

Marker	Description	How to Display Parameter in User Message
%d	Decimal Number	as decimal number
%x	Hexadecimal Number	as hexadecimal number
%L	Slot Label Code	as text description of Slot Label Code

Table B-2: Status Message ID Definitions

Status Message ID	Value	Data Value 1	Data Value 2	Status ID Description
STS_CAL_FAIL	0x0001	Slot Label Code		"%L failed calibration"
STS_SENS_NOT_FOUND	0x0002	Slot Label Code		"%L sensor not found"
STS_SENS_ALWAYS_ON	0x0003	Slot Label Code		"%L sensor always on"
STS_LAMP_DOUSED	0x0011	N/A		"Lamp doused"
STS_LAMP_STRIKE	0x0012	N/A		"Lamp failed to strike"
STS_OVERTEMP	0x0021	Sensor Number	Temperature	"Sensor %d over temp at %d degrees C"
STS_UNDERTEMP	0x0022	Sensor	Temperature	"Sensor %d under temp at %d degrees C"
STS_SENS_OUT_RANGE	0x0023	Sensor Number		"Sensor %d out of range"
STS_OVERVOLTAGE_	0x0031	Phase Number	Voltage	"Phase %d over voltage at

PHASE				%d V."
STS_UNDERVOLTAGE_PHASE	0x0032	Phase Number	Voltage	"Phase %d under voltage at %d V."
STS_OVERCURRENT	0x0033	Phase Number	Current	"Phase %d over current at %d A."
STS_UNDERCURRENT	0x0034	Phase Number	Current	"Phase %d under current at %d A."
STS_PHASE	0x0035	Phase Number	Phase Angle	"Phase %d is at %d degrees"
STS_PHASE_ERROR	0x0036	Phase Number		"Phase %d Error."
STS_AMPS	0x0037	Current		"%d Amps"
STS_VOLTS	0x0038	Volts		"%d Volts"
STS_DIMSLOT_OCCUPIED	0x0041	N/A		"No Dimmer"
STS_BREAKER_TRIP	0x0042	N/A		"Tripped Breaker"
STS_WATTS	0x0043	Wattage		"%d Watts"
STS_DIM_FAILURE	0x0044	N/A		"Dimmer Failure"
STS_DIM_PANIC	0x0045	N/A		"Panic Mode"
STS_READY	0x0050	Slot Label Code		"%L ready"
STS_NOT_READY	0x0051	Slot Label Code		"%L not ready"
STS_LOW_FLUID	0x0052	Slot Label Code		"%L low fluid"

Appendix C: Slot Info (Normative)

Table C-1 defines the available Slot Types. Table C-2 defines some publicly available Slot IDs. Informative note: The ESTA website (<http://www.esta.org/rdm>) may document new publicly defined Slot IDs in addition to those enumerated in this document. Implementers of this standard are encouraged to check the ESTA website for applicable Slot IDs before creating manufacturer-specific codes.

Manufacturers should, whenever possible, use publicly defined Slot IDs rather than developing their own Manufacturer-specific IDs.

Publicly defined IDs are in the range of 0x0000 - 0x7FFF. Manufacturer-specific IDs are in the range of 0x8000 – 0xFFDF.

The Slot IDs are organized into categories to provide logical groupings, which controllers can use to provide a more intuitive user interface for Manufacturer-specific codes.

Table C-1: Slot Type

Slot ID Type	Value	Description
ST_PRIMARY	0x00	Slot directly controls parameter (represents Coarse for 16-bit parameters)
ST_SEC_FINE	0x01	Fine, for 16-bit parameters
ST_SEC_TIMING	0x02	Slot sets timing value for associated parameter
ST_SEC_SPEED	0x03	Slot sets speed/velocity for associated parameter
ST_SEC_CONTROL	0x04	Slot provides control/mode info for parameter
ST_SEC_INDEX	0x05	Slot sets index position for associated parameter
ST_SEC_ROTATION	0x06	Slot sets rotation speed for associated parameter
ST_SEC_INDEX_ROTATE	0x07	Combined index/rotation control
ST_SEC_UNDEFINED	0xFF	Undefined secondary type

Table C-2: Slot ID Definitions

Slot Category/ID	Value	Description
<i>Intensity Functions</i>	<i>0x00xx</i>	
SD_INTENSITY	0x0001	Intensity
SD_INTENSITY_MASTER	0x0002	Intensity Master
<i>Movement Functions</i>	<i>0x01xx</i>	
SD_PAN	0x0101	Pan
SD_TILT	0x0102	Tilt
<i>Color Functions</i>	<i>0x02xx</i>	
SD_COLOR_WHEEL	0x0201	Color Wheel
SD_COLOR_SUB_CYAN	0x0202	Subtractive Color Mixer – Cyan/Blue
SD_COLOR_SUB_YELLOW	0x0203	Subtractive Color Mixer – Yellow/Amber
SD_COLOR_SUB_MAGENTA	0x0204	Subtractive Color Mixer - Magenta

SD_COLOR_ADD_RED	0x0205	Additive Color Mixer - Red
SD_COLOR_ADD_GREEN	0x0206	Additive Color Mixer - Green
SD_COLOR_ADD_BLUE	0x0207	Additive Color Mixer - Blue
SD_COLOR_CORRECTION	0x0208	Color Temperature Correction
SD_COLOR_SCROLL	0x0209	Color Scroll
SD_COLOR_SEMAPHORE	0x0210	Color Semaphore
<i>Image Functions</i>	<i>0x03xx</i>	
SD_STATIC_GOBO_WHEEL	0x0301	Static gobo wheel
SD_ROTO_GOBO_WHEEL	0x0302	Rotating gobo wheel
SD_PRISM_WHEEL	0x0303	Prism wheel
SD_EFFECTS_WHEEL	0x0304	Effects wheel
<i>Beam Functions</i>	<i>0x04xx</i>	
SD_BEAM_SIZE_IRIS	0x0401	Beam size iris
SD_EDGE	0x0402	Edge/Lens focus
SD_FROST	0x0403	Frost/Diffusion
SD_STROBE	0x0404	Strobe/Shutter
SD_ZOOM	0x0405	Zoom lens
SD_FRAMING_SHUTTER	0x0406	Framing shutter
SD_SHUTTER_ROTATE	0x0407	Framing shutter rotation
SD_DOUSER	0x0408	Douser
SD_BARN_DOOR	0x0409	Barn Door
<i>Control Functions</i>	<i>0x05xx</i>	
SD_LAMP_CONTROL	0x0501	Lamp control functions
SD_FIXTURE_CONTROL	0x0502	Fixture control channel
SD_FIXTURE_SPEED	0x0503	Overall speed setting applied to multiple or all parameters
SD_MACRO	0x0504	Macro control
SD_UNDEFINED	0xFFFF	No definition

Appendix D: Definitions (Normative)

D.1 Acknowledge: a response to a request that indicates if the request was successful, and if not, why the request failed.

D.2 Alternate START Code (ASC): a START Code with a non-zero value.

D.3 Binary search or binary tree search: a method of searching for a value in a range of values by continually halving or limiting the search range. A diagram of such a search resembles a tree.

D.4 Bidirectional Half Duplex: in a Bidirectional Half Duplex data link, requests flowing one direction and responses flowing the opposite direction are all carried over the same transmission media. This differs from a bidirectional full duplex data link in which requests flow over one transmission media, and responses flow over a second transmission media.

D.5 Bit Distortion: changes in bit timing due in part to unsymmetrical propagation delays and transition times. Bit distortion can be caused by both active electronics and passive circuit elements.

D.6 Bus Release: switching off a port transceiver's transmitter.

D.7 Byte: an unsigned, 8-bit value that can represent the numbers ranging from 0 to 255.

D.8 Checksum: an aid in verifying that data is successfully transmitted and received. In RDM communication, the checksum field is the unsigned, modulo 0x10000, 16-bit additive checksum of the entire packet's slot data, including START Code.

D.9 Collision or data collision: a data collision occurs when two or more devices attempt to transmit data at the same time on the same data link. The most likely outcome of this collision is that the data will become garbled. With the exception of the discovery process, RDM communication relies upon polling to avoid data collisions.

D.10 Controller: in any segment of an RDM communication system, only one device is free to initiate data transmissions. This device is termed a Controller.

D.11 Command port: a port from which RDM requests originate.

D.12 Controller Transmitter: the transmitter on a controller.

D.13 Data Delay: the time difference between the time data enters a circuit element and the time the data leaves the circuit element.

D.14 Data Link: the physical connection between transmitting and receiving devices.

D.15 Destination UID: the unique ID of the device to which a message is being sent.

D.16 Device: a piece of electrical or electronic equipment attached to an RDM communication system capable of transmitting, receiving and/or passing RDM messages.

D.17 Discovery: the process of finding RDM devices in an RDM communication system.

D.18 Driver Disable Time: the amount of time from the End of Packet to the time when the transmitter stops driving the data link.

D.19 Driver Enable Time: the amount of time from when a device begins driving the data link to the time when a device actually begins transmitting data onto the data link.

D.20 Duty Cycle: During a period of time, the fraction of time the signal is at a high level.

D.21 Encoded Unique ID (EUID): to help prevent data collisions during the Discovery process from being interpreted as NSC packets, device UID's are encoded in a fashion that insures there are no consecutive 0's in a device response. The result of this encoding is termed an Encoded Unique ID.

D.22 End of Packet (EOP): the end of the second stop bit of the last byte of the RDM packet. For the Unique Branch response packet this is the end of the second stop bit of the last byte of the EUID. For all other RDM packets, this is the end of the second stop bit of the last byte of the checksum.

D.23 Broadcast Device UID: a means of addressing a single message to a group of devices.

D.24 Gnd: an abbreviation for electrical ground, or the electrical power supply output from which other power supply outputs are measured or referenced.

D.25 In-line Devices: refers to devices that receives and re-transmits DMX512.

D.26 Inter-slot Delay: the amount of time between the end of one slot and beginning of the next slot in the packet.

D.27 Line Bias Network: the pull-up & pull-down circuits that create the marking bias when the RS485 line is not driven (all drivers are in high impedance mode).

D.28 Line Termination: the termination resistors located at both ends of a physical DMX512 link that are used to match the line impedance.

D.29 Manufacturer ID (MID): a two-byte value assigned to a Manufacturer/Organization by the E1 Accredited Standards Committee for use with specific Alternate START Codes. This ID identifies the data contained in the packet as proprietary. This packet may be safely ignored by systems that do not implement the specific start code.

D.30 Mark: a line condition where Data+ is high with respect to Data-. A Mark represents a binary 1.

D.31 Line Bias: a voltage potential impressed between Data+ and Data- when the data link is not driven (all drivers are in high impedance mode) that causes a Mark to be sensed by receiving devices.

D.32 Marking State: a Marking State exists when the voltage levels on the data+ and data- wires of the data link cause a logic 1 to be sensed by the various port receivers.

D.33 Mute: during the discovery process, devices are located by asking them to respond if their UID is within the range of UID's contained in the Discovery Unique Branch message. Once a device has been found, the device is instructed to stop responding to the unique number queries.

The Mute request is used to instruct the device to stop responding. A device that has been instructed to stop responding is said to be "muted".

D.34 NULL START Code (NSC): a START Code with a value of zero (00h).

D.35 Packet: all the electrical transitions on the data link necessary to send a complete message is termed a Packet. In the context of RDM communications, a packet is comprised of a break, mark after break, message header, message data (if any), and checksum.

D.36 Parameter: a device attribute about which requests are sent and responses are made.

D.37 Polling: to prevent complete chaos on the data link, devices may only transmit data in response to a request from the controller. Polling is this process of request and response.

D.38 Port Turnaround: the switching of a port from the transmitting state to a receiving state, or from a receiving state to a transmitting state.

D.39 Proxy Device: any in-line device that acts as an agent or representative for one or more devices.

D.40 Responder Port: the port from which RDM responses originate.

D.41 Root Device: the top level device in a piece of equipment containing, either physically or logically, sub-devices. An example of a root device is the top level rack controller in a dimmer rack.

D.42 Segment: in its simplest form, a DMX512 network consists of a controller and controlled devices with no in-line devices. The insertion of in-line devices into the network divides the network into Segments.

D.43 Slot: a sequentially numbered, framed byte within the RDM packet.

D.44 Source UID: the unique ID of the device sending a message.

D.45 Space: a line condition where Data+ is low with respect to Data-. A Space represents a binary 0.

D.46 Splitter: A splitter is a device that receives a DMX/RDM data stream and replicates it on multiple command ports. RDM responses are replicated only on the responder port.

D.47 START Code: the first slot sent after the break indicating the type of information to follow.

D.48 Start of Packet (SOP): the leading edge of the BREAK at the beginning of an RDM packet. In the case of breakless responses (discovery), the SOP is defined as the leading edge of the start bit of the first slot sent by the responder.

D.49 Sub-device: a device contained, either physically or logically, in another device. Individual sub-devices do not have UID's, but are referred to by using the root device's UID. A dimmer that is part of a dimmer rack is an example of a sub-device.

D.50 Sub-START Code: the second byte in the RDM packet. The sub-START Code is intended to allow future expansion of the START Code interface. Currently, the sub-START Code is

assigned a default value.

D.51 Termination: components placed at the ends of a transmission line to match the impedance of the transmission line, preventing reflection of the data signals back into the transmission line that would degrade or destroy the transmitted signals.

D.52 Transaction Number: the Transaction Number is an unsigned 8-bit field. Controller generated packets increment this field every time a packet is transmitted. This field shall be initialized to 0 and roll over from 255 to 0. Responders echo the Transaction Number from the Controller Packet. Responders do not independently generate Transaction Numbers.

D.53 Unique ID (UID): a number used to uniquely identify a device.

D.54 Vcc: represents the power supply voltage used to power an electrical or electronic circuit.

Appendix E: Discovery Pseudo-Code Example (Informative)

E.1 Find Devices Function Call

Find_Devices(0, 0x000000FFFFFFFFFFFFE); // main call for device discovery process.

E.2 Find Devices Pseudo-Code

```
bool Find_Devices(long long LowerBound, long long UpperBound,)
{
    long long MidPosition;
    int DeviceFound;

    if (LowerBound == UpperBound) // if we have called to the lowest branch and are testing
                                   // for a single device.
    {
        do
        {
            Send DISC_MUTE Command to UID of device at 'LowerBound'.

        } while( no DISC_MUTE responses && <10 attempts);

        if ( response received )
        {
            Add device found at 'LowerBound' to list.
        }
        else
            return( true );
    }
    else // look for a lower branch with active devices.
    {
        do
        {
            Send DISC_UNIQUE_BRANCH message.

        } while ( <3 attempts && no response)

        if ( response received )
        {
            DeviceFound = true;

            // check for single good response.
            // !!! remove during FULL DEBUGGING VALIDATION begin
            if ( response checksum passes ) // possibly only one device in branch.
            {
                // shortcut to avoid branching all the way down.
                DeviceFound = Quick_Find( );
            }
        }
    }
}
```

```

    }
    // !!! remove during FULL DEBUGGING VALIDATION end

    //choose next branches to test
    if (DeviceFound == true ) // Quick_Find indicates there are multiple
        // devices in branch.
    {
        MidPosition = ((LowerBound & (0x0000800000000000-1)) +
            (UpperBound & (0x0000800000000000-1))) / 2
        + ((UpperBound & 0x0000800000000000) ? 0x0000400000000000 : 0)
        + ((LowerBound & 0x0000800000000000) ? 0x0000400000000000 : 0);

        //go into next branch fork.
        DeviceFound = Find_Devices (MidPosition + 1, UpperBound);
        DeviceFound |= Find_Devices (LowerBound, MidPosition);

        if (DeviceFound)
            return( true );
    }
}
return( false );
}

```

```

bool Quick_Find( )
{
    //The call to this sub-function should be removed during full discovery testing as it can
    //mask other //problems in the discovery implementation.

    // This sub-function can speed up discovery when there is only one device in the current
    // branch by // eliminating the need to fully branch to the lowest level of the discovery tree.
    do
    {
        Send DISC_MUTE Command to Source UID of Device from Response Message.

    } while( no DISC_MUTE responses && <10 attempts);

    if ( response received )
        Add Source UID of device found to list.

    do // verify there is no other devices in this branch.
    {
        Send DISC_UNIQUE_BRANCH message.

    } while ( <3 attempts && no response)

    if ( response && checksum passes)
    {

```

```
        // there is another single device response at this branch.
        Quick_Find();
    }
    else if ( response && failed checksum) // there are possibly multiple devices...
                                           // branch further
        return( true );
    else // there is nothing left at this branch...move on.
        return( false );
}
```

Appendix F: Qualification tests for transmitter/receiver circuits used in RDM systems (Normative)

F.1 Qualification Tests for Command port Transmitter Circuits

The combination of the line bias network of Section 2.5 and an EIA485 transmitter may present greater than a single unit load. The combination of the line biasing network and the transmitter design shall be tested for conformance with this standard and EIA485. The test circuits for command ports are given in Figure F-1 and Figure F-2.

Command port designs that pass these tests shall be considered capable of driving the command port load and 31 additional unit loads.

Responder port transmitters will also be affected by the line biasing network. These Responder ports are tested using the circuit shown in Figure F-4.

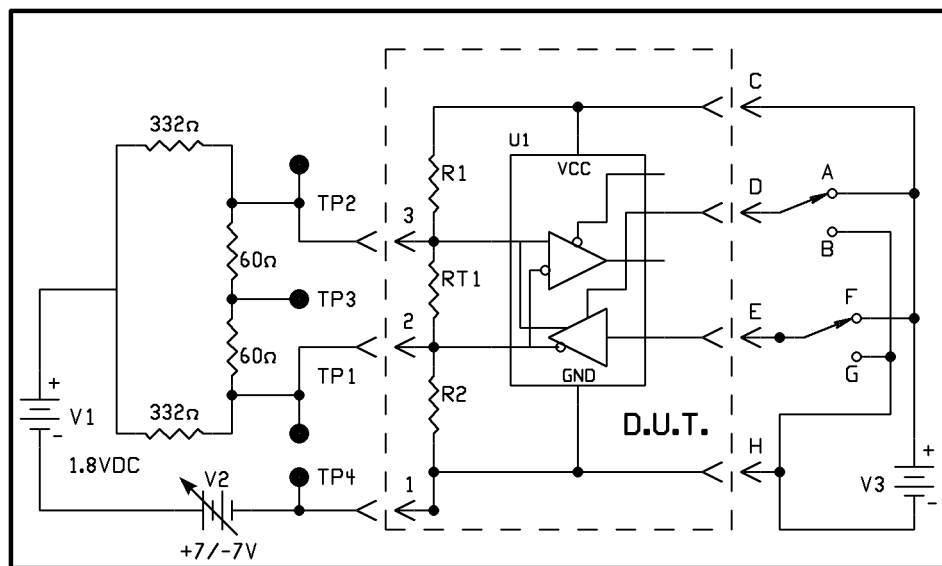


Figure F-1: Command Port Transmitter Test Circuit

F.1.1 Notes on Figure F-1

The Device Under Test (DUT) consists of the command port line driver, the line biasing network used in the design and any additional components used between the driver and the connection point. The DUT in Figure F-1 uses the reference line biasing network for illustration only.

The 1.8 volt reference (V1) and the variable reference (V2) (shown as batteries) shall be low impedance voltage sources capable of both sinking and sourcing current. The DUT shall be powered using the same voltage as the production circuit (V3). A method to switch the transmitter data input to either logic 1 or 0 shall be provided. (switch F/G). A method to disable (place in high impedance mode) or enable the transmitter is provided (Switch A/B). Resistors used to build the test load network shall be within 1% of stated values.

F.2 Output tests for testing transmitters / line bias networks for RDM command ports

Tests for verifying transmitter circuits:

1) With the DUT transmitter disabled (placed in a high impedance state) the common mode voltage supply (V2) is varied from -7 VDC to +7 VDC. The bias voltage between TP1 and TP2 is observed and shall be greater than or equal to 245 mV for all allowed values of the common mode voltage.

2) With the DUT transmitter enabled the common mode supply is varied from -7 VDC to +7 VDC. The output between TP1 and TP2 is observed. The output shall have magnitude of greater than or equal to 1.5 VDC for all allowed values of common mode voltage and for either setting of switch F/G. It should be noted that 1.5 VDC is a minimum value; greater magnitudes are desirable.

3) Set the common mode supply to zero. The magnitude of the output between TP1 and TP2 shall vary less than 200 mV for either setting of switch F/G. The magnitude of the voltage difference between TP3 and TP4 shall be less than 200mV for either setting of switch F/G.

F.3 Line loading tests for command ports

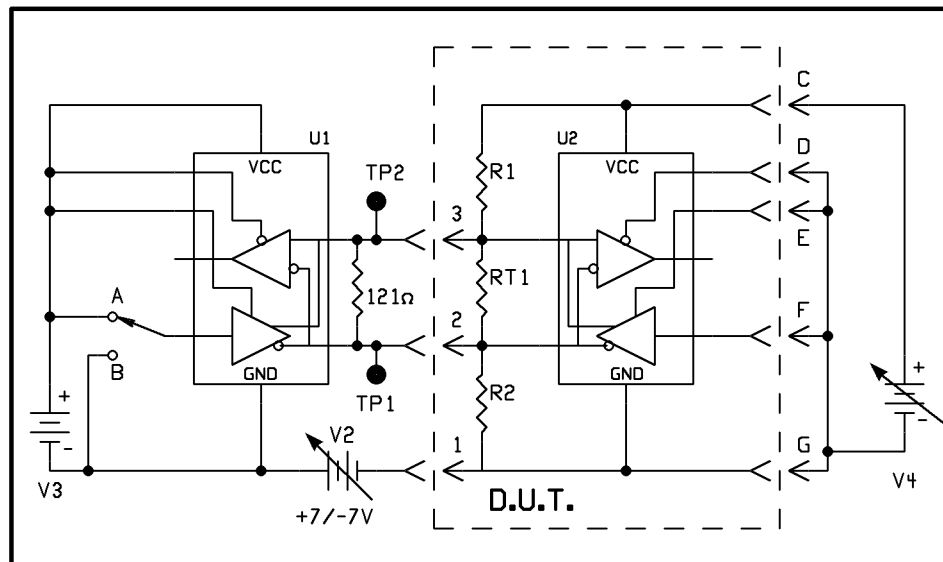


Figure F-2: Command Port Load Test Circuit

The following tests shall be used to determine that line biasing networks correctly load responder transmitters on the data link. This is achieved by comparison of the calibration circuit to the DUT.

The test circuit shown in Figure F-2 consists of an EIA485 transceiver (U1), with transmitter enabled, and powered by an appropriate supply (V3). The test circuit provides a means to switch the polarity of its output between mark and space. The test consists of two parts.

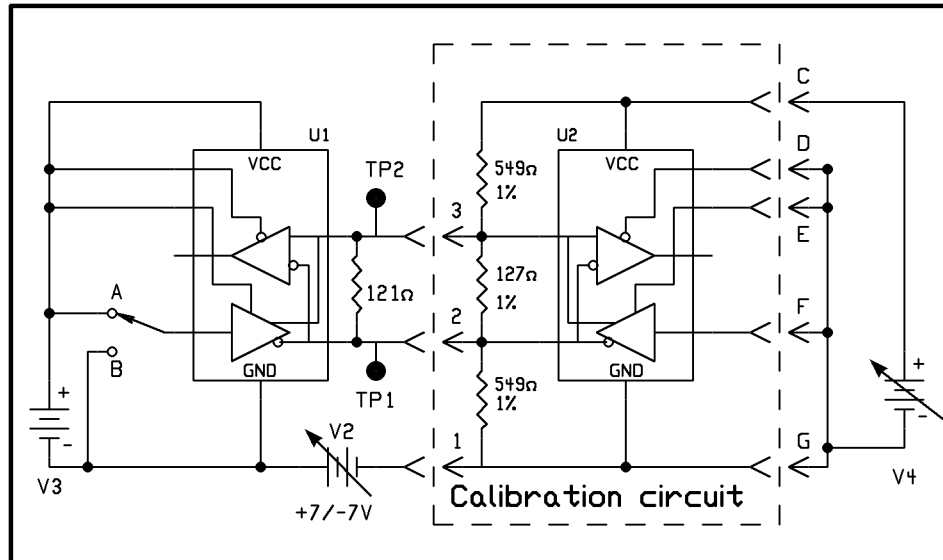


Figure F-3: Calibration Circuit

Test1: Substitute the calibration circuit shown in Figure F-3 for the DUT. Set the supply for U2 (V4) and the line bias network to 5.5 volts. Set the output of U1 to Mark. Measure the voltage between TP1 and TP2, while adjusting the common mode supply (V2) between +7VDC and -7VDC. Record the lowest magnitude measured.

Switch the output of U1 to space. While adjusting the common mode supply between +7VDC and -7VDC continue to measure the voltage between TP1 and TP2. Record the lowest magnitude measured.

Test2: Connect the DUT to the test circuit. Adjust the power supply (V4) to the high tolerance specified for the DUT. Set the output of U1 to Mark. Measure the voltage between TP1 and TP2, while adjusting the common mode supply between +7VDC and -7VDC. Record the lowest magnitude measured.

The DUT conforms to this standard if the lowest magnitude recorded for the DUT is greater than or equal to the lowest magnitude recorded for the calibration circuit.

F.4 Notes on the responder test circuit

The DUT shown in Figure F-4 consists of the responder line driver and any additional components used between the driver and the connection point.

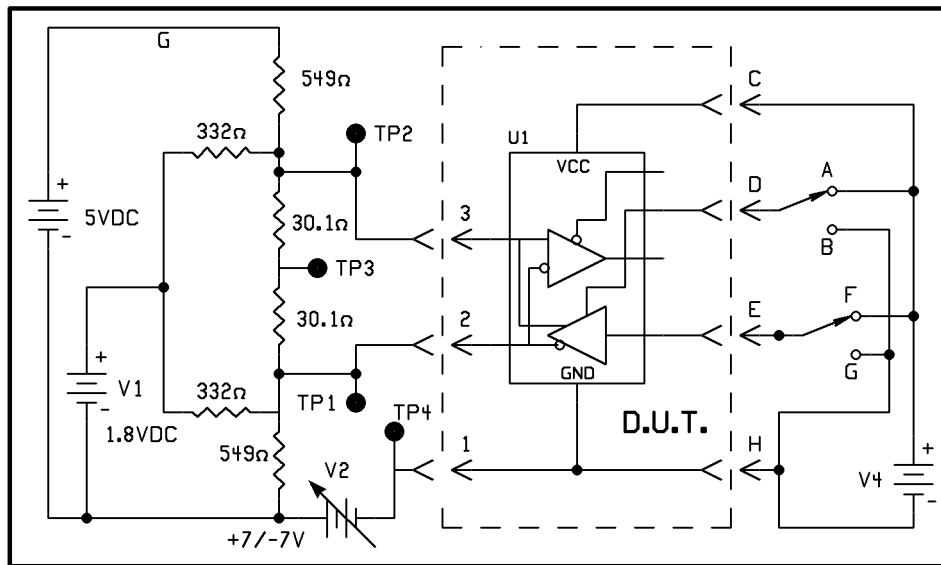


Figure F-4: Responder Test Circuit

The 1.8 volt reference (V1) and the variable reference (V2) (shown as batteries) shall be low impedance voltage sources capable of both sinking and sourcing current. The power supply for the test load (G) shall be 5 volts DC. A method to switch the transmitter data input to either logic 1 or 0 shall be provided (switch F/G). A method to either disable (place in a high impedance state) or enable the transmitter is provided (Switch A/B). Resistors used to build the test load network shall be within 1% of stated values. V4 shall be set to equal the supply voltage that will run the DUT in production.

F.5 Testing Responder Transmitters

With the DUT transmitter enabled, and with either setting of switch E/F, the common mode supply (V2) is varied between -7 VDC and +7 VDC. The output magnitude shall be greater than or equal to 1.5 VDC for all values of common mode voltage.

Set the common mode supply (V2) to zero. The magnitude of the output between TP1 and TP2 shall vary less than 200 mV for either setting of switch F/G. The magnitude of the voltage difference between TP3 and TP4 shall be less 200mV for either setting of switch F/G.

Responder port designs that pass these tests shall be considered capable of driving the command port load and 31 additional unit loads.

Contact Information

E1 Secretariat

Entertainment Services and Technology Association

Karl G. Ruling

Technical Standards Manager

ESTA

875 Sixth Avenue, Suite 1005

New York, NY 10001

Phone: +1-212-244-1505

FAX: +1-212-244-1502

email: kruling@esta.org

Technical Standards Committee Chairperson

Mr. Mike Garl

President

James Thomas Engineering, Inc.

10240 Caneel Drive

Knoxville, TN 37931

Phone: +1-865-692-3060

FAX: +1-865-692-9020

email: mikeg@jthomaseng.com

Control Protocols Working Group Chairpersons

Tracy Underhill

Electronics Diversified Inc.

1675 N.W. Cornelius Pass Road

Hillsboro, OR 97124

Phone: +1-503-645-5533

FAX: +1-503-629-9877

email: tracy.underhill@edionline.com

Steve Terry

Electronic Theatre Controls, Inc.

630 Ninth Avenue, Suite 1001

New York, NY 10036

Phone: +1-212-397-8080

FAX: +1-212-397-4340

email: sterry@etcconnect.com

Acknowledgments:

Control Protocols Working Group - RDM Task Group Members:

Chair / Editor: Scott M. Blair, High End Systems / Flying Pig Systems (USA)

Javid Butler, Integrated Theatre (USA)
Milton Davis, Doug Fleenor Design (USA)
Gary Douglas, Pathway Connectivity (Canada)
Doug Fleenor, Doug Fleenor Design (USA)
Bob Goddard, Goddard Design (USA)
Tom Grimes, High End Systems (USA)
Wayne Howell, Artistic Licence (UK)
Kevin Loewen, Pathway Connectivity (Canada)
Jason Potterf, High End Systems / Flying Pig Systems (USA)
Charles Reese, Production Resource Group (USA)
Michael A. (Sandy) Twose, Sand Network Systems (Canada)
Tracy Underhill, Electronics Diversified Inc. (USA)
Peter Willis, Howard Eaton Lighting Ltd. (UK)

Control Protocols Working Group - Principal voting members at the time this document was approved by the Working Group on January. 21. 2006:

Arnold Tang; Arnold Tang Productions [U]
Wayne David Howell; Artistic Licence (UK) Ltd. [CP]
Tobin Neis; Barbizon Companies [DR]
Doug Fleenor; Doug Fleenor Design, Inc. [MP]
Daniel W. Antonuk; Electronic Theatre Controls, Inc. [MP]
Tracy Underhill; Electronics Diversified Inc. [MP]
Ralph Weber; ENDL Texas [G]
Philip Nye; Engineering Arts [G]
Robert Goddard; Goddard Design Co. [MP]
Scott M. Blair; High End Systems Inc. [MP]
Alan Martello; Horizon Control Inc. [MP]
Peter Willis; Howard Eaton Lighting Ltd. [CP]
Edwin S. Kramer; I.A.T.S.E. Local 1 [U]
Roger Lattin; I.A.T.S.E. Local 728 [U]
John (Javid) D. Butler; Integrated Theatre, Inc. [CP]
Rick Leinen; Leviton Manufacturing Co., Inc. [MP]
Torben Kaas Rasmussen; Martin Professional A/S [G]
George Kindler; Network Installation Corporation [DR]
Gary Douglas; Pathway Connectivity Inc. [MP]
Nic Bowker; PLASA [G]
Charles Reese; Production Resource Group [DR]
Hans Lau; Sand Network Systems [MP]
Richard Lawrence; Strand Lighting Ltd. [MP]
Mitch Hefter; USITT [U]
John Sondericker III; Wybron, Inc. [MP]

Alternate voting members:

Simon Hobday; Artistic Licence (UK) Ltd. [CP]
Milton Davis; Doug Fleenor Design, Inc. [MP]
Ken Wagner; Doug Fleenor Design, Inc. [MP]
Steve Terry; Electronic Theatre Controls, Inc. [MP]
Jason Potterf; High End Systems Inc. [MP]
Tom Grimes; High End Systems, Inc. [MP]
Robert Bell; Horizon Control Inc. [MP]
John Huntington; I.A.T.S.E. Local 1 [U]
Dennis Grow; I.A.T.S.E. Local 728 [U]
Alan M. Rowe; I.A.T.S.E. Local 728 [U]
Michael (Mike) Whetstone; Integrated Theatre, Inc. [CP]
Ken Vannice; Leviton Manufacturing Co., Inc. [MP]
Flemming Jensen; Martin Professional A/S [G]
Dave Higgins; Pathway Connectivity Inc. [MP]
Kevin Loewen; Pathway Connectivity Inc. [MP]
Michael A. (Sandy) Twose; Sand Network Systems [MP]
Yngve Sandboe; Sand Network Systems [MP]
Michael Lay; Strand Lighting Ltd. [MP]

Observer members:

Jean-Francois Canuel; A.C. Lighting Ltd. [CP]
Tim Bachman; A.C.T Lighting, Inc. [DR]
Andre Broucke; ADB-TTV Group [MP]
John Sellers; AIM Northwest [G]
Steve Friedlander; Auerbach Pollock Friedlander [U]
J. B. Toby; Avolites Ltd. [MP]
Shahid Anwar; Avolites Ltd. [MP]
Richard Salzedo; Avolites Ltd. [MP]
Barbara Wohlsen; Barbara Wohlsen [U]
Jiantong Wu; Beijing Special Engineering Design &
Research Institute [G]
Bernardo Benito Rico; Ben-Ri Electronica S.A. [MP]
Lee J. Bloch; Bloch Design Group, Inc. [G]
Soo-Myong Chung; Blue Sky Design, Ltd. [G]
Bradley Klinkradt; Bradley Klinkradt [G]
Bill Ellis; Candela Controls, Inc. [U]
Will Wagner; Carallon Ltd. [MP]
Steve Roberts; Carr & Angier [G]
Marty Lazarus; Chicago Spotlight, Inc. [G]
Chuck Seifried; City of Phoenix [U]
Larry Dunn; City Theatrical, Inc. [CP]
Fabiano Pina; Clay Paky S.P.A. [MP]

Observer members: continued

Ohad Ashery; Compulite Systems [MP]
Simeon Aladjem; Compulite Systems [MP]
Yehuda Shukram; Compulite Systems [MP]
Dietmar Rottinghaus; Connex GmbH [G]
David Bertenshaw; David Bertenshaw [G]
Larry Schoeneman; Designlab Chicago, Inc. [DR]
Gary Dove; Dove Systems [MP]
Jerry Durand; Durand Interstellar, Inc. [CP]
Jussi Kallioinen; Eastway Sound & Lighting [U]
Edward R. Condit; Edward R. Condit [G]
Ed Jones; Edwin Jones Co., Inc. [CP]
Joost van Eenbergen; ELC Lighting [MP]
Bill Fehrmann; Electrol Engineering, Inc. [MP]
Hans Leiter; Electronic Theatre Controls, Inc. [MP]
Anders Ekvall; Electronic Theatre Controls, Inc. [MP]
Adam Bennette; Electronic Theatre Controls, Inc. [MP]
Erwin Rol; Erwin Rol [G]
Sierk Janszen; Ground Zero [U]
Liao Wei Min; GuangZhou HeDong
Electronic Co., Ltd. [G]
Paul J. Clark; Hacek Design Ltd. [CP]
Alan P. Symonds; Harvard University [U]
Trevor Forrest; Helvar Lighting Control [MP]
Bill Hewlett; Hewlett Electronics [CP]
Steve Carlson; High Speed Design, Inc. [MP]
Gian Carlo C. Bartolotti; Ibeam SP/Banco de Eventos [U]
Geoffrey O. Thompson; IEEE 802.3/Nortel Networks [G]
Mark Jensen; Intelligent Control Devices [CP]
Rob Johnston; Interactive Technologies, Inc. [MP]
David Timmins; Jands Electronics [MP]
Helge Hoffmann; JB Lighting [MP]
Edward Paget; Jones & Phillips Associates, Inc. [G]
Christopher Purpura; Jones & Phillips
Associates, Inc. [G]
Lawrence Neal; Lawrence Neal [CP]
John Mehlretter; Lehigh Electric Products Co. [MP]
Mark T. Kraft; Lehigh Electric Products Co. [MP]
Lars Wernlund; Lewlight [MP]
Klaus Amling; Licht-Technik; P
Andrew Sherar; Lightmoves PLC [MP]
Avraham Mendall Mor "Avi"; Lightswitch [U]
Simon Alpert; Lighttech Event Technologies [CP]
Gary Pritchard; LSC Lighting Systems PTY Ltd [MP]
Bart Swinnen; Luminex LCE [MP]
J. P. Steiner; Lutron Electronics [MP]
Jim Holladay; Luxence [G]

Daniel Weiermann; Mainstage Theatrical Supply [DR]
Hiroshi Kita; Marumo Electric Co., Ltd. [MP]
Petri Laine; Obelux Oy [CP]
Stuart Cotts; Oregon Shakespeare Festival [U]
David A. Boller; Organic Machines LLC [CP]
William Benner; Pangolin Laser Systems [MP]
Greg Reimer; Pathway Connectivity Inc. [MP]
James Eade; PLASA [G]
Mac Perkins; PNTA Inc. [G]
Paul F. Mardon; Pulsar Ltd. [MP]
Steve Unwin; Pulsar Ltd. [MP]
Stephen J. Tyrrell; Quantum Logic [MP]
Douglas Franz; QVC Network [U]
Sean Harding; Rhode Island College [U]
Charlie Richmond; Richmond Sound Design Ltd. [CP]
Martin Farnik; Robe Show Lighting s.r.o. [MP]
Kenneth Mockler; Sceno Plus [U]
Mick Martin; ShowCAD Control Systems [MP]
Loren Wilton; Showman Systems [CP]
Jon R. Farley; Sixteenth Avenue Systems [CP]
Robert Barbagallo; Solotech Inc. [U]
Michael Gonzales; Spectrum Lighting Inc. [DR]
Ujjal Kar; Standard Robotics & Lighting [G]
Paul K. Ericson; Syska Hennessy Group
Lighting Design [U]
Stephen Bickford; T. Kondos Associates [U]
Tad Trylski; Tad Trylski [U]
Klas Dalbjorn; TC Group [MP]
Pete Baselici; The Watt Stopper [MP]
Jerry Gorrell; Theatre Safety Programs [G]
Stéphane Jacob; TIR Systems Ltd. [MP]
Colin Waters; TMB [U]
Torrey Bievenour; Vision Quest Lighting [G]
Eckart Steffens; VPLT [G]
Eric Cornwell; West Side Systems [U]

ANSI E1.20 – 2006, Entertainment Technology – RDM - Remote Device Management over DMX512 Networks
CP/2003-1003r4 -137-

Entertainment Services and
Technology Association



American National Standard
E1.27-1 - 2006
Entertainment Technology
Standard for Portable Control Cables
for Use with ANSI E1.11 (DMX512-A) and
USITT DMX512/1990 Products

Entertainment Services and Technology Association



American National Standard E1.27-1 - 2006 Entertainment Technology Standard for Portable Control Cables for Use with ANSI E1.11 (DMX512-A) and USITT DMX512/1990 Products CP/2003-1028r5.1

This edition of ANSI E1.27 was approved by American National Standards Institute on June 6 , 2006.

©2006 ASC E1, Safety and Compatibility of Entertainment Technical Equipment and Practices, and its secretariat the Entertainment Services and Technology Association. All rights reserved. No part of this publication may be reproduced in any material form (including photocopying or storing by electronic means) without the written permission of the copyright holder. Any parties wishing to translate and publish this document in another language must receive permission from the copyright holder.

Notice and Disclaimer

ESTA and ANSI Accredited Standards Committee E1 (for which ESTA serves as the secretariat) do 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, or an American National Standard developed under Accredited Standards Committee E1 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 or Accredited Standards Committee E1.

ESTA and ANSI Accredited Standards Committee E1 (ASC E1) neither guaranty nor warrant 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 and ASC E1 do 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 judgement 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**

875 Sixth Avenue, Suite 1005
New York, NY 10001 USA
Phone: +1-212-244-1505
Fax: +1-212-244-1502
Email: standards@esta.org

For Electronic Copies of this Document Contact:

www.estafoundation.org

For Additional Copies of this Document Contact:

ESTA Publications

The ESTA Foundation
875 Sixth Avenue, Suite 1005
New York, NY 10001 USA
Phone: +1-212-244-1505
Fax: +1-212-244-1502

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, PLASA, 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 as Accredited Standards Committee E1, Safety and Compatibility of Entertainment Technical Equipment and Practices.

The Technical Standards Committee (TSC) was established by ESTA's Board of Directors to oversee and coordinate the Technical Standards Program. Made up of individuals experienced in standards-making work from throughout our industry, the Committee approves all projects undertaken and assigns them to the appropriate working group. The Technical Standards Committee employs a Technical Standards Manager to coordinate the work of the Committee and its working groups as well as maintain a "Standards Watch" on behalf of members. Working groups include: Camera Cranes, Control Protocols, Electrical Power, Floors, Fog and Smoke, Photometrics, and Rigging.

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 two 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 manufacturers, consultants, dealers, and end-users. ESTA is committed to developing consensus-based standards and recommended practices in an open setting. Future Control Protocols Working Group projects will include updating this publication as changes in technology and experience warrant, as well as developing new standards and recommended practices for the benefit of the entertainment industry.

Contents

Foreword	1
1 General	2
1.1 Scope	2
1.2 Overview and Architecture	2
1.3 Compliance	2
2 Normative References	2
3 Definitions	3
4 Electrical Characteristics	4
4.1 Background	4
4.2 Maximum and Minimum Cable Lengths	4
4.3 Construction	4
4.4 Impedance	5
4.5 Capacitance	5
4.6 Dielectric Withstanding Protection	5
5 Connection Methods	5
5.1 Required Connector	5
6 Electrical Specifications and Physical Layer	5
6.1 General	5
6.2 DMX512 Portable Cables	5
6.3 Data Link Common and Grounding Topologies	6
6.4 Data Link Signal Designations Summary	6
7 Required Portable Cable Disclosures	6

Foreword

(This foreword contains no mandatory requirements and is not part of E1.27-1)

This standard describes the types of portable cable used to interconnect products which comply with ANSI E1.11, Entertainment Technology – USITT DMX512-A: Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories.

In 2003, The Control Protocols Working Group of ESTA's Technical Standards Program authorized the formation of a DMX512 Cabling Task Group. Writing an American National Standard for the use of portable DMX512-A cables was one of the projects assigned to this Task Group.

This document is the result. It has been developed under the Policies and Procedures of the ESTA Technical Standards Program.

Task Group Acknowledgements:

Task Group Chair: John David Butler, Integrated Theatre, Inc.

Editor: Ted Paget, Jones & Phillips Associates, Inc.

Co-Editor: Chris Purpura, Jones & Phillips Associates, Inc.

Tim Bachman, ACT

Milton Davis, Doug Fleenor Design

Mitch Hefter, Entertainment Technology

Peter Willis, Howard Eaton Lighting Limited

1 General

1.1 Scope

This Standard describes the types of portable cable for the transmission of digital data among products which comply with ANSI E1.11, Entertainment Technology – USITT DMX512-A. It covers recommended cable types, connectors and their internal wiring.

This Standard is intended as a guide for:

1. Equipment manufacturers and system specifiers who wish to integrate systems of lighting equipment and accessories, including dimmers, with controllers made by different manufacturers.
2. System specifiers and consultants who wish to gain detailed information about recommended cable types and allowed connectors.

This standard is intended to supplement ANSI E1.11, Entertainment Technology — USITT DMX512-A — Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories. References to ANSI E1.11, and DMX512-A within this standard all refer to this document.

This standard is not intended to replace existing portable digital data cabling standards or recommended practices other than those described in USITT DMX512 and DMX512/1990.

Unless otherwise noted, references to DMX512 in this document refer to DMX512-A.

1.2 Overview and Architecture

The means of transport of DMX512-A digital data from one compliant device to another is normally a two-pair cable, with each pair serving as a data link. Single pair cables are allowed when properly marked to differentiate them from two-pair cables. Portable cable shall be shielded to protect the data links from interference (RFI and EMI). The physical connection of portable cables at any device is via a 5-pin XLR connector.

The first pair of wires in any DMX512 portable data cable is used as the primary data link. The second pair is used for a variety of purposes, all of which fall within the scope of DMX512-A.

1.3 Compliance

Compliance with this Standard is strictly voluntary and the responsibility of the manufacturer. Disclosures and identification or other claims of compliance do not constitute certification or approval by the E1 Accredited Standards Committee. See clause 7 for Disclosure requirements.

2 Normative References

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

ANSI/TIA/EIA-568-B-2001 *Commercial Building Telecommunications Cabling Standard*

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

USITT DMX512/1990 Digital Data Transmission Standard for Dimmers and Controllers
USITT
6443 Ridings Rd.
Syracuse, NY 13206-1111
+1-800-938-7488 +1-315-463-6463 Fax: +1-315-463-6525
<http://www.usitt.org>

3 Definitions

3.1 Circuit Common: the common reference (zero volt supply) of the EIA-485-A driver or receiver circuitry.

3.2 Common: see Data Link, Signal Common, and Circuit Common.

3.3 Data+: signal true.

3.4 Data-: signal complement.

3.5 Data Link: The physical connection between transmitting and receiving devices.

3.6 Data Link Common: the connection to circuit Common at the point of interconnection (DMX512 Port) of the product.

3.7 Declaration: Declared in a manual and optionally marked on the device.

3.8 DMX512 Port: a DMX512 signal connection point (connector or terminal strip).

3.9 Earth Ground: the common, zero potential available from the mains electricity supply and usually connected to the metal chassis of equipment. Earth Ground is referred to as Earth in Europe and Ground in the USA.

3.10 EMI: Electromagnetic Interference.

3.11 Legacy (as used in this Standard): transmitting and receiving devices complying with the original USITT DMX512 or DMX512/1990 in all aspects of those standards. (Exception: receiving devices that are not dimmers but comply with all other aspects of DMX512/1990 shall be considered to be Legacy Equipment.)

3.12 RFI: Radio Frequency Interference.

3.13 Receiver (Receiving Device): a piece of equipment that accepts a DMX512 signal.

3.14 Signal Common: the common reference conductor of the physical media (e.g., the cable shield).

3.15 Transmitting Device: a piece of equipment that produces a DMX512 signal.

4 Electrical Characteristics

4.1 Background

The data transmission rate (250 kbits/s) used by DMX512 requires the selection of a portable DMX512 cable that does not significantly distort the signal or give rise to spurious signal reflections. Cables intended for use with audio systems (such as microphone cables), while having the convenience of flexibility, availability, and relative low cost, may not be suitable for use with DMX512 because of their high capacitance and incorrect characteristic impedance; at DMX512 data rates this will give rise to bit time distortion and signal reflections/overshoot.

4.2 Maximum and Minimum Cable Lengths

Maximum and minimum run lengths are specifically omitted from this standard. A number of factors, including signal quality, device operating characteristics including capacitive values, and installation environment can affect these lengths, and is beyond the scope of this standard.

4.3 Construction

Portable DMX512 cables shall use twisted pair conductors. Conductors shall be of stranded construction. The raw cable used for a DMX512 cable assembly shall be declared by its manufacturer as suitable for use with EIA-422/EIA-485/EIA-485-A systems. Shielding shall be on individual pairs or overall shielding of pairs, or both. The portable cable itself shall be flexible and rugged enough for the repeated coiling and uncoiling to which it will be subjected.

4.3.1 Cables implementing only the Primary Data Link shall consist of at least one twisted pair and be marked according to Clause 7.1.

4.3.2 Cables implementing both Data Links shall consist of at least two twisted pairs and be marked according to Clause 7.1.

4.3.3 Cables implementing only the Secondary Data Link shall not be allowed.

4.4 Impedance

Portable DMX512 cables shall have a characteristic impedance in the range 100 to 120 ohms. Due to the characteristic impedance of 120 ohms in EIA-485 systems, 120 ohms is preferred.

4.5 Capacitance

Capacitance between conductors within a shield shall not exceed 19.8 pF/ft (65 pF/m). Capacitance between any conductor and the shield shall not exceed 35 pF/ft (115 pF/m).

4.6 Dielectric Withstanding Protection

Dielectric rating for portable DMX512 cables shall conform to prevailing electrical codes. Requirements for specific voltage ratings, insulation types, jacket materials and other characteristics vary with location and application, and are beyond the scope of this Standard.

5 Connection Method

5.1 Required Connector

Portable cables shall use 5-pin XLR connectors. The physical pin designations of the 5-pin XLR shall be in accordance with Table 1.

Any use of alternate connectors shall comply with ANSI E1.11.

6 Electrical Specifications and Physical Layer

6.1 General

This standard addresses portable cables for use in DMX512 systems that conform to EIA-485-A and additional physical layer requirements. Where a conflict exists, DMX512-A shall govern. The physical layer of a DMX512 data link is constrained by earth grounding practices, termination methods, signal levels, EMI, and accidental damage by connection to other devices.

6.2 DMX512 Portable Cables

6.2.1 General

A DMX512 Portable Cable is a digital data transmission cable designed for the provisional interconnection of two DMX512 devices. Portable cables shall each have two prescribed connectors, a male 5-pin XLR at the end nearest the transmitting device and a female 5-pin XLR at the end nearest the receiving device. Pins shall be designated 1 through 5. There shall be no connection to the shell.

6.2.2 Compatibility with Legacy Equipment/Adaptors

Some legacy equipment placed voltages on the second pair of data conductors that may damage other connected DMX512 devices. Portable adaptors intended to break the second pair for the purpose of protecting DMX512 devices are beyond the scope of this standard.

Adaptors such as turnarounds, gender changers, taps, etc. are beyond the scope of this standard.

Users are cautioned that E1.27-1 cables may be connected to adaptors that change the functionality of the E1.27-1 cable.

6.3 Data Link Common and Grounding Topologies

In all cases Pin 1 of DMX512 portable cable connectors shall act as Data Link Common. The wire connected to Pin 1 shall be no smaller than the wire used for the twisted pairs in the cable.

6.4 Data Link Signal Designations Summary

Table 1 - Signal Designations Summary

Use	5-Pin XLR Pin	DMX512 Function
Common Reference	1	Data Link Common
Primary Data Link	2	Data 1 -
	3	Data 1 +
Secondary Data Link	4	Data 2 -
	5	Data 2 +

Each data link shall consist of a separate twisted pair.

7 Required Portable Cable Disclosures

7.1 Portable DMX512 cables shall come with the following declaration: “Complies with ANSI E1.27-1 – Standard for Portable Control Cables for Use with ANSI E1.11 (DMX512-A) and USITT DMX512/1990 Products”.

7.2 The cable assembly shall be marked with “ANSI E1.27-1” at both ends of the cable. Cables shorter than 6.6 feet (2m) long shall be permitted to be labeled at one end only.

7.3 Cables which implement both Data Links shall be additionally marked with the text, “Two Pair” or “2-Pair”.

7.4 Cables which implement only the Primary Data Link shall be additionally marked with the text, “Single Pair” or “1-Pair”. Such cables shall also be marked with a violet colored band a minimum of 1/2-inch (12.7 mm) wide around the entire circumference of the cable jacket within 2 inches (50.8 mm) of the required text markings.

7.5 All marks shall be made in a durable manner.

Contact Information

E1 Secretariat

Entertainment Services and Technology Association

Karl G. Ruling

Technical Standards Manager

ESTA

875 Sixth Avenue, Suite 1005

New York, NY 10001

Phone: +1-212-244-1505

FAX: +1-212-244-1502

email: kruling@esta.org

Technical Standards Committee Chairperson

Mr. Mike Garl

President

James Thomas Engineering, Inc.

10240 Caneel Drive

Knoxville, TN 37931

Phone: +1-865-692-3060

FAX: +1-865-692-9020

email: mikeg@jthomaseng.com

Control Protocols Working Group Chairpersons

Tracy Underhill

Electronics Diversified Inc.

1675 N.W. Cornelius Pass Road

Hillsboro, OR 97124

Phone: +1-503-645-5533

FAX: +1-503-629-9877

email: tracy.underhill@edionline.com

Steve Terry

Electronic Theatre Controls, Inc.

630 Ninth Avenue, Suite 1001

New York, NY 10036

Phone: +1-212-397-8080

FAX: +1-212-397-4340

email: sterry@etcconnect.com

Control Protocols Working Group

Principal voting members at the time this document was approved by the Working Group on January, 21, 2006:

Arnold Tang; Arnold Tang Productions [U]
Wayne David Howell; Artistic Licence (UK) Ltd. [CP]
Tobin Neis; Barbizon Companies [DR]
Doug Fleenor; Doug Fleenor Design, Inc. [MP]
Daniel W. Antonuk; Electronic Theatre Controls, Inc. [MP]
Tracy Underhill; Electronics Diversified Inc. [MP]
Ralph Weber; ENDL Texas [G]
Philip Nye; Engineering Arts [G]
Robert Goddard; Goddard Design Co. [MP]
Scott M. Blair; High End Systems Inc. [MP]
Alan Martello; Horizon Control Inc. [MP]
Peter Willis; Howard Eaton Lighting Ltd. [CP]
Edwin S. Kramer; I.A.T.S.E. Local 1 [U]
Roger Lattin; I.A.T.S.E. Local 728 [U]
John (Javid) D. Butler; Integrated Theatre, Inc. [CP]
Rick Leinen; Leviton Manufacturing Co., Inc. [MP]
Torben Kaas Rasmussen; Martin Professional A/S [G]
George Kindler; Network Installation Corporation [DR]
Gary Douglas; Pathway Connectivity Inc. [MP]
Nic Bowker; PLASA [G]
Charles Reese; Production Resource Group [DR]
Hans Lau; Sand Network Systems [MP]
Richard Lawrence; Strand Lighting Ltd. [MP]
Mitch Hefter; USITT [U]
John Sondericker III; Wybron, Inc. [MP]

Alternate voting members:

Simon Hobday; Artistic Licence (UK) Ltd. [CP]
Milton Davis; Doug Fleenor Design, Inc. [MP]
Ken Wagner; Doug Fleenor Design, Inc. [MP]
Steve Terry; Electronic Theatre Controls, Inc. [MP]
Jason Potterf; High End Systems Inc. [MP]
Tom Grimes; High End Systems, Inc. [MP]
Robert Bell; Horizon Control Inc. [MP]
John Huntington; I.A.T.S.E. Local 1 [U]
Dennis Grow; I.A.T.S.E. Local 728 [U]
Alan M. Rowe; I.A.T.S.E. Local 728 [U]
Michael (Mike) Whetstone; Integrated Theatre, Inc. [CP]
Ken Vannice; Leviton Manufacturing Co., Inc. [MP]
Flemming Jensen; Martin Professional A/S [G]
Dave Higgins; Pathway Connectivity Inc. [MP]
Kevin Loewen; Pathway Connectivity Inc. [MP]
Michael A. (Sandy) Twose; Sand Network Systems [MP]
Yngve Sandboe; Sand Network Systems [MP]
Michael Lay; Strand Lighting Ltd. [MP]

Observer members:

Jean-Francois Canuel; A.C. Lighting Ltd. [CP]
Tim Bachman; A.C.T Lighting, Inc. [DR]
Andre Broucke; ADB-TTV Group [MP]
John Sellers; AIM Northwest [G]
Steve Friedlander; Auerbach Pollock Friedlander [U]
J. B. Toby; Avolites Ltd. [MP]
Shahid Anwar; Avolites Ltd. [MP]
Richard Salzedo; Avolites Ltd. [MP]
Barbara Wohlsen; Barbara Wohlsen [U]
Jiantong Wu; Beijing Special Engineering Design & Research Institute [G]
Bernardo Benito Rico; Ben-Ri Electronica S.A. [MP]
Lee J. Bloch; Bloch Design Group, Inc. [G]
Soo-Myong Chung; Blue Sky Design, Ltd. [G]
Bradley Klinkrad; Bradley Klinkrad [G]
Bill Ellis; Candela Controls, Inc. [U]
Will Wagner; Carallon Ltd. [MP]
Steve Roberts; Carr & Angier [G]
Marty Lazarus; Chicago Spotlight, Inc. [G]
Chuck Seifried; City of Phoenix [U]
Larry Dunn; City Theatrical, Inc. [CP]
Fabiano Pina; Clay Paky S.P.A. [MP]
Ohad Ashery; Compulite Systems [MP]
Simeon Aladjem; Compulite Systems [MP]
Yehuda Shukram; Compulite Systems [MP]
Dietmar Rottinghaus; Connex GmbH [G]
David Bertenshaw; David Bertenshaw [G]
Larry Schoeneman; Designlab Chicago, Inc. [DR]
Gary Dove; Dove Systems [MP]
Jerry Durand; Durand Interstellar, Inc. [CP]
Jussi Kallioinen; Eastway Sound & Lighting [U]
Edward R. Condit; Edward R. Condit [G]
Ed Jones; Edwin Jones Co., Inc. [CP]
Joost van Eenbergen; ELC Lighting [MP]
Bill Fehrmann; Electrol Engineering, Inc. [MP]
Hans Leiter; Electronic Theatre Controls, Inc. [MP]
Anders Ekvall; Electronic Theatre Controls, Inc. [MP]
Adam Bennette; Electronic Theatre Controls, Inc. [MP]
Erwin Rol; Erwin Rol [G]
Sierk Janszen; Ground Zero [U]
Liao Wei Min; GuangZhou HeDong Electronic Co., Ltd. [G]
Paul J. Clark; Hacek Design Ltd. [CP]
Alan P. Symonds; Harvard University [U]
Trevor Forrest; Helvar Lighting Control [MP]
Bill Hewlett; Hewlett Electronics [CP]
Steve Carlson; High Speed Design, Inc. [MP]
Gian Carlo C. Bartolotti; Ibeam SP/Banco de Eventos [U]
Geoffrey O. Thompson; IEEE 802.3/Nortel Networks [G]
Mark Jensen; Intelligent Control Devices [CP]
Rob Johnston; Interactive Technologies, Inc. [MP]
David Timmins; Jands Electronics [MP]
Helge Hoffmann; JB Lighting [MP]
Edward Paget; Jones & Phillips Associates, Inc. [G]
Christopher Purpura; Jones & Phillips Associates, Inc. [G]
Lawrence Neal; Lawrence Neal [CP]
John Mehlretter; Lehigh Electric Products Co. [MP]
Mark T. Kraft; Lehigh Electric Products Co. [MP]
Lars Wernlund; Lewlight [MP]
Klaus Amling; Licht-Technik; P
Andrew Sherar; Lightmoves PLC [MP]
Avraham Mendall Mor "Avi"; Lightswitch [U]
Simon Alpert; Lighttech Event Technologies [CP]
Gary Pritchard; LSC Lighting Systems PTY Ltd [MP]
Bart Swinnen; Luminex LCE [MP]
J. P. Steiner; Lutron Electronics [MP]
Jim Holladay; Luxence [G]
Daniel Weiermann; Mainstage Theatrical Supply [DR]
Hiroshi Kita; Marumo Electric Co., Ltd. [MP]
Petri Laine; Obelux Oy [CP]
Stuart Cotts; Oregon Shakespeare Festival [U]
David A. Boller; Organic Machines LLC [CP]
William Benner; Pangolin Laser Systems [MP]
Greg Reimer; Pathway Connectivity Inc. [MP]
James Eade; PLASA [G]
Mac Perkins; PNTA Inc. [G]
Paul F. Mardon; Pulsar Ltd. [MP]
Steve Unwin; Pulsar Ltd. [MP]
Stephen J. Tyrrell; Quantum Logic [MP]
Douglas Franz; QVC Network [U]
Sean Harding; Rhode Island College [U]
Charlie Richmond; Richmond Sound Design Ltd. [CP]
Martin Farnik; Robe Show Lighting s.r.o. [MP]
Kenneth Mockler; Sceno Plus [U]
Mick Martin; ShowCAD Control Systems [MP]
Loren Wilton; Showman Systems [CP]
Jon R. Farley; Sixteenth Avenue Systems [CP]
Robert Barbagallo; Solotech Inc. [U]
Michael Gonzales; Spectrum Lighting Inc. [DR]
Ujjal Kar; Standard Robotics & Lighting [G]
Paul K. Ericson; Syska Hennessy Group Lighting Design [U]
Stephen Bickford; T. Kondos Associates [U]
Tad Trylski; Tad Trylski [U]
Klas Dalbjorn; TC Group [MP]
Pete Baselici; The Watt Stopper [MP]
Jerry Gorrell; Theatre Safety Programs [G]
Stéphane Jacob; TIR Systems Ltd. [MP]
Colin Waters; TMB [U]
Torrey Bievenour; Vision Quest Lighting [G]
Eckart Steffens; VPLT [G]
Eric Cornwell; West Side Systems [U]

[CP] = custom-market producer [DR] = dealer rental company [G] = general interest
[MP] = mass-market producer [U] = user

Entertainment Services and
Technology Association



American National Standard
E1.31- 2009
Entertainment Technology –
Lightweight streaming protocol for transport of
DMX512 using ACN

[inside front cover]

Entertainment Services and Technology Association



American National Standard E1.31 — 2009 Entertainment Technology Lightweight streaming protocol for transport of DMX512 using ACN

Revision 0.46 23 October 2008
CP/2006-1020r3

This document was approved as an American National Standard by the ANSI Board of Standards Review
on 4 May 2009.

© 2009 The Entertainment Services and Technology Association. All rights reserved.

This page intentionally blank

Notice and Disclaimer

The Entertainment Services and Technology Association 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 an American National Standard developed by ESTA 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
875 Sixth Avenue, Suite 1005
New York, NY 10001 USA
Phone: 1-212-244-1505
Fax: 1-212-244-1502
standards@esta.org
<http://www.esta.org/>

For additional copies of this document contact:

ESTA Publications
The ESTA Foundation
875 Sixth Avenue, Suite 1005
New York, NY 10001 USA
Phone: 1-212-244-1505
Fax: 1-212-244-1502
<http://www.estafoundation.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 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, PLASA, 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 Committee (TSC) was established by ESTA's Board of Directors to oversee and coordinate the Technical Standards Program. Made up of individuals experienced in standards-making work from throughout our industry, the Committee approves all projects undertaken and assigns them to the appropriate working group. The Technical Standards Committee employs a Technical Standards Manager to coordinate the work of the Committee and its working groups as well as maintain a "Standards Watch" on behalf of members. Working groups include: Camera Cranes, Control Protocols, Electrical Power, Floors, Fog and Smoke, Followspot Position, Photometrics, and Rigging.

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, 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 also can become involved by requesting that the TSC develop a standard 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 in an open setting. Future Control Protocols Working Group projects will include updating this publication as changes in technology and experience warrant, as well as developing new standards for the benefit of the entertainment industry.

Contact Information

Entertainment Services and Technology Association

Karl G. Ruling
Technical Standards Manager
ESTA
875 Sixth Avenue, Suite 1005
New York, NY 10001
Phone: 1-212-244-1505
FAX: 1-212-244-1502
standards@esta.org

Technical Standards Committee Chairperson

Mike Garl
James Thomas Engineering, Inc.
10240 Caneel Drive
Knoxville, TN 37931
Phone: 1-865-692-3060
FAX: 1-865-692-9020
mikeg@jthomaseng.com

Control Protocols Working Group Chairpersons

Michael Lay
Strand Lighting
6603 Darin Way
Cypress, CA 90630
Phone: 1-714-230-8208
FAX: 1-714-899-0042
mlay@strandlight.com

Tracy Underhill
Electronics Diversified LLC
1675 NW Cornelius Pass Rd.
Hillsboro, OR 97124
Phone: 1-503-645-5533
FAX: 1-503-629-9877
tracy.underhill@edionline.com

Acknowledgments

The Control Protocols Working Group was the consensus body for the development of this standard. The working group's membership at the time the working group approved this standard on 1 January 2009 is listed below.

Voting members:

Daniel W. Antonuk; Electronic Theatre Controls, Inc. [MP]
Robert Armstrong; Pathway Connectivity Inc. [MP]
Robert Bell; Horizon Control Inc. [MP]
Scott M. Blair; High End Systems Inc. [MP]
Ron Bonner; PLASA [G]
John (Javid) D. Butler; Integrated Theatre, Inc. [CP]
Jean-Francois Canuel; Spectrum Manufacturing Inc. [CP]
Kimberly Corbett; Schuler Shook [G]
Stuart Cotts; Oregon Shakespeare Festival [U]
Milton Davis; Doug Fleenor Design, Inc. [MP]
Gary Douglas; Horizon Control Inc. [MP]
Doug Fleenor; Doug Fleenor Design, Inc. [MP]
Robert Goddard; Goddard Design Co. [MP]
Tom Grimes; High End Systems, Inc. [MP]
Dennis Grow; I.A.T.S.E. Local 728 [U]
Mitch Hefter; Entertainment Technology, representing USITT [U]
Dave Higgins; Pathway Connectivity Inc. [MP]
Simon Hobday; Artistic Licence (UK) Ltd. [CP]
John Huntington; City Tech, Ent Tech Department, representing I.A.T.S.E. Local 1 [U]
Edwin S. Kramer; I.A.T.S.E. Local 1 [U]
Ulrich Kunkel; E3 Engineering & Education for Entertainment, representing DIN [U]
Roger Lattin; I.A.T.S.E. Local 728 [U]
Hans Lau; Sand Network Systems, Inc. [MP]
Michael Lay; Strand Lighting [MP]
Rick Leinen; Leviton Manufacturing Co., Inc. [MP]
Kevin Loewen; Pathway Connectivity Inc. [MP]
Mark Manthei; Shure Inc. [G]
Alan Martello; Horizon Control Inc. [MP]
Tyrone Mellon Jr.; Lex Products Corp. [CP]
Philip Nye; Engineering Arts [G]
Edward A. (Ted) Paget; Vortek Rigging Division of Daktronics Inc. [G]
Charles Reese; Production Resource Group [DR]
Alan M. Rowe; IATSE Local 728 [U]
Yngve Sandboe; Sand Network Systems, Inc. [MP]
Arnold Tang; Arnold Tang Productions [U]
Steve Terry; Electronic Theatre Controls, Inc. [MP]
Robert Tooker; Production Resource Group [DR]
Tracy Underhill; Electronics Diversified LLC [MP]
Ken Vannice; Leviton Manufacturing Co., Inc. [MP]
Michael (Mike) Whetstone; Integrated Theatre, Inc. [CP]
Peter Willis; Howard Eaton Lighting Ltd. [CP]
Kehang Wu; Shure Inc. [G]

Observer members:

Simon Alpert; Lighttech Event Technologies [CP]
Klaus Amling; Licht-Technik; P
Shahid Anwar; Avolites Ltd. [MP]
Tim Bachman; A.C.T Lighting, Inc. [DR]
Robert Barbagallo; Solotech Inc. [U]
William Benner; Pangolin Laser Systems [MP]
Adam Bennette; Electronic Theatre Controls Ltd. [MP]
David Bertenshaw; David Bertenshaw [G]
Stephen Bickford; T. Kondos Associates [U]
Torrey Bievenour; Vision Quest Lighting [G]
Mike Blackwell; Philips/Color Kinetics [MP]
Lee J. Bloch; Bloch Design Group, Inc. [G]
David A. Boller; Organic Machines LLC [CP]
Andre Broucke; ADB - TTV Technologies [MP]
Steve Carlson; High Speed Design, Inc. [MP]
Soo-Myong Chung; Bloch Design Group, Inc. [G]
Paul J. Clark; HxDx [CP]
Edward R. Condit [G]
Eric Cornwell; West Side Systems [U]
Klas Dalbjorn; Labgruppen AB [MP]
Wayne David Howell; Artistic Licence (UK) Ltd. [CP]
Gary Dove; Dove Systems [MP]
Larry Dunn; City Theatrical, Inc. [CP]
Jerry Durand; Durand Interstellar, Inc. [CP]
James Eade; PLASA [G]
Joost van Eenbergen; ELC Lighting [MP]
Anders Ekvall; Transtechnik Lichtsysteme GmbH [MP]
Bill Ellis; Candela Controls, Inc. [U]
Paul K. Ericson; Syska Hennessy Group Lighting Design [U]
Jon R. Farley; Sixteenth Avenue Systems [CP]
Martin Farnik; Robe Show Lighting s.r.o. [MP]
Bill Fehrmann; Electrol Engineering, Inc. [MP]
Trevor Forrest; Helvar Lighting Control [MP]
Douglas Franz; QVC Network [U]
Steve Friedlander; Auerbach Pollock Friedlander [U]
Michael Gonzales; Spectrum Lighting Inc. [DR]
Jerry Gorrell; Theatre Safety Programs [G]
Josh Gubler [CP]
Sean Harding; Rhode Island College [U]
Bill Hewlett; Hewlett Electronics [CP]
Helge Hoffmann; JB Lighting [MP]
Jim Holladay; Luxence [G]
Sierk Janszen; Kiss Box [U]
Flemming Jensen; Martin Professional A/S [G]
Eric Johnson [MP]
Rob Johnston; Interactive Technologies, Inc. [MP]
Ed Jones; Edwin Jones Co., Inc. [CP]
Jussi Kallioinen; Eastway Sound & Lighting [U]
Ujjal Kar; Standard Robotics & Lighting [G]

Hiroshi Kita; Marumo Electric Co., Ltd. [MP]
 Mark T. Kraft; Lehigh Electric Products Co. [MP]
 Marty Lazarus; Chicago Spotlight, Inc. [G]
 Hans Leiter; ETC GmbH [MP]
 Paul F. Mardon; Pulsar Ltd. [MP]
 Mick Martin; ShowCAD Control Systems [MP]
 Paul Kenneth McEwan; Zero 88 Lighting Limited [MP]
 John Mehlretter; Lehigh Electric Products Co. [MP]
 Avraham Mendall Mor "Avi"; Lightswitch [U]
 Tobin Neis; Barbizon Companies [DR]
 Lars F. Paape; Scientific Algorithms and Embedded Systems [U]
 Fabiano Pina; Clay Paky S.P.A. [MP]
 Gary Pritchard; LSC Lighting Systems (Aust) PTY Ltd [MP]
 Eric Proces [U]
 Christopher Purpura; Jones & Phillips Associates, Inc. [G]
 Torben Kaas Rasmussen; Martin Professional A/S [G]
 Charlie Richmond; Richmond Sound Design Ltd. [CP]
 Bernardo Benito Rico; Ben-Ri Electronica S.A. [MP]
 Steve Roberts; Carr & Angier [G]
 Erwin Rol; Erwin Rol Software Engineering [G]
 Dietmar Rottinghaus; Connex GmbH [G]
 Richard Salzedo; Avolites Ltd. [MP]
 Larry Schoeneman; Designlab Chicago, Inc. [DR]
 Chuck Seifried; Phoenix Civic Plaza [U]
 John Sellers; AIM Northwest [G]
 Andrew Sherar; Lightmoves PLC [MP]
 Yehuda Shukram; Compulite Systems [MP]
 John Sondericker III; Wybron, Inc. [MP]
 Bart Swinnen; Luminex LCE [MP]
 Geoffrey O. Thompson; Nortel Networks, Inc. [G]
 David Timmins; Jands Electronics [MP]
 J. B. Toby; Avolites Ltd. [MP]
 Bob Toms; Catalyst Microsystems LLC [G]
 Tad Trylski; Tad Trylski [U]
 Stephen J. Tyrrell; Quantum Logic [MP]
 Steve Unwin; Pulsar Ltd. [MP]
 Dominic Vincenty; TPS [DR]
 Will Wagner; Carallon Ltd. [MP]
 John Warwick; Philips/Color Kinetics [MP]
 Colin Waters; TMB [U]
 Ralph Weber; ENDL Texas [G]
 Daniel Weiermann; Mainstage Theatrical Supply - Milwaukee [DR]
 Lars Wernlund; Lewlight [MP]
 Loren Wilton; Showman Systems [CP]
 Barbara Wohlsen [U]
 Jiantong Wu; Beijing Special Engineering Design & Research Institute [G]

[CP] Custom-market Producer
 [MP] Mass-market Producer
 [DR] Dealer or Rental company

[U] User
 [G] General Interest

Table of Contents

Notice and Disclaimer.....	i
Acknowledgments	iv
Table of Contents	vii
1 Introduction.....	1
1.1 Scope	1
1.2 Overview and Architecture	1
1.3 Appropriate use of this standard	1
1.4 Classes of data appropriate for transmission.....	1
1.5 Compliance.....	1
2 Normative References.....	1
3 Definitions.....	3
4 Protocol Packet Structure Summary	3
5 E1.31 use of the ACN Root Layer Protocol.....	5
5.1 Preamble Size	5
5.2 Post-amble Size	5
5.3 ACN Packet Identifier	5
5.4 Flags & Length	5
5.5 Vector	5
5.6 CID (Component Identifier)	5
6 E1.31 Framing Layer Protocol.....	6
6.1 Flags & Length	6
6.2 Vector	6
6.3 Source Name.....	6

6.4 Priority	6
6.4.1 Multiple Sources at Highest Priority	6
6.4.2 Discussion of Merge and Arbitration Algorithms	7
6.4.3 Discussion of Resolution of Sources Exceeded Condition	7
6.4.4 Requirements for Merging and Arbitrating	7
6.4.5 Requirements for <i>Sources Exceeded</i> Resolution	7
6.4.6 Requirements for Devices with Multiple Operating Modes	8
6.5 Sequence Number.....	8
6.6 Options	8
6.7 Universe	8
6.8 Framing Layer Operation and Timing - Transmitter Requirements	8
6.8.1 Transmission Rate.....	8
6.8.2 Null START Code Transmission Requirements.....	9
6.9 Framing Layer Operation and Timing - Receiver Requirements	9
6.9.1 Network Data Loss	9
6.9.2 Sequence Numbering.....	9
7 DMP Layer Protocol	9
7.1 Flags & Length	10
7.2 Vector	10
7.3 Address Type and Data Type.....	10
7.4 First Property Address.....	10
7.5 Address Increment	10
7.6 Property Value Count	10
7.7 Property Values (DMX512-A Data)	10
8 Operation of E1.31 in IPv4 Networks	11

8.1 Assignment of IP Addresses	11
8.2 Association of Multicast Addresses and Universe	11
8.3 Multicast Subscription.....	11
8.4 Allocation of Multicast Addresses.....	11
9 Translation between DMX512-A and E1.31	12
9.1 DMX512-A to E1.31 Translation.....	12
9.1.1 Boot Condition	12
9.1.2 Temporal Sequence	12
9.1.3 Loss of Data	12
9.2 E1.31 to DMX512-A Translation.....	12
9.2.1 General.....	12
9.2.2 Loss of Data	12

1 Introduction

1.1 Scope

This standard describes a mechanism to transfer DMX512-A [DMX] packets over a TCP/IP network using a subset of the ACN protocol suite. It covers data format, data protocol, data addressing, and network management.

1.2 Overview and Architecture

This standard can be used to transfer DMX512-A [DMX] packets of all START Codes via an ANSI E1.17 [ACN] supported network. A simple packet wrapper approach is used whereby the DMX512-A [DMX] data is encapsulated in a wrapper following the ACN packet structure. The ACN standard wrapper is carried in UDP [UDP] packets when used on TCP/IP networks. In the future, this use of the ACN wrapper and packet structure will also allow E1.31 to be carried over other networks supported by ACN.

The wrapper is structured such that it is both compatible and meaningful to the ANSI E1.17 [ACN] standard. Readers are referred to the ANSI E1.17 [ACN] standard, particularly the “ACN Architecture” and “Device Management Protocol” documents for more information. The “Root Layer Protocol” used in this standard is described in the “ACN Architecture” document.

This standard uses multicast addressing to provide a mechanism to partition traffic for distinct universes of DMX512-A [DMX] data. Direct unicast of DMX512-A [DMX] data is also supported.

1.3 Appropriate use of this standard

This standard uses a non-reliable transport mechanism to stream packets of data from multiple controllers to multiple receivers over the ACN network. Like DMX512-A [DMX] over EIA-485 media, there is no acknowledgement and therefore no assurance that all packets have been received.

1.4 Classes of data appropriate for transmission

This standard, E1.31, is intended to define a method to carry DMX512-A [DMX] style data over IP Networks, including Ethernet IP Networks. It is designed to carry repetitive control data from one or more controllers to one or more receivers. This protocol is intended to be used to control dimmers, other lighting devices, and related nonhazardous effects equipment.

1.5 Compliance

Compliance with this standard is strictly voluntary and the responsibility of the implementer. Markings and identification or other claims of compliance do not constitute certification or approval by the E1 accredited standards committee.

2 Normative References

[DMX] ANSI E1.11 Entertainment Technology – USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for controlling lighting equipment and accessories.

This standard is maintained by ESTA.

ESTA
875 Sixth Avenue, Suite 1005
New York, NY 10001
+1-212-244-1505
<http://www.esta.org>

ESTA is a standardization body accredited by ANSI to develop, maintain, and withdraw American National Standards.

ANSI
25 West 43rd Street
4th floor
New York, NY 10036
+1-212-642-4900
<http://www.ansi.org>

[ACN] ANSI E1.17 Entertainment Technology – Architecture for Control Networks

This standard is maintained by ESTA.

[RDM] ANSI E1.20 Entertainment Technology – Remote Device Management over DMX512 networks.

This standard is maintained by ESTA.

[UTF8] The Unicode Consortium. The Unicode Standard, Version 5.0.0, defined by:
The Unicode Standard, Version 5.0 (Boston, MA, Addison-Wesley, 2007. ISBN 0-321-48091-0)

[UDP] RFC 0768 UDP User Datagram Protocol

This standard is maintained by the IETF.
IETF Secretariat
c/o NeuStar, Inc.
Corporate Headquarters
46000 Center Oak Plaza
Sterling, VA 20166
Phone: +1-571-434-3500
www.ietf.org

[IGMP2] RFC 2236 IGMPv2 Internet Group Management Protocol Version 2.

This standard is maintained by the IETF.

[IGMP3] RFC 3376 Internet Group Management Protocol, Version 3

This standard is maintained by the IETF.

[ASIPM] RFC 2365 Administratively Scoped IP Multicast.

This standard is maintained by the IETF.

[UUID] RFC 4122 P. Leach, M. Mealling, and R. Salz. *A Universally Unique IDentifier (UUID) URN Namespace*. July 2005.

This standard is maintained by the IETF.

3 Definitions

3.1 Octet: An eight bit byte within a data packet.

3.2 Universe: A set of up to 512 data slots identified by universe number. Note: In E1.31 there may be multiple sources for a universe. See also Slot.

3.3 Universe Number: Each packet contains a universe number identifying the universe it carries. From an ACN perspective, a receiving device has some number of properties whose value is addressed by the combination of a universe number and a data slot number. From an historical perspective, a receiving device consumes some number of DMX512 [DMX] data slots.

3.4 Slot: A slot is a sequentially numbered octet in a DMX512 [DMX] packet. A single Universe contains a maximum of 513 Slots, starting at slot 0. Slot 0 is the DMX512 [DMX] START Code. Slots 1 through 512 are data slots.

3.5 Source: A stream of data packets for a universe is said to be sent from a *source*. A *source* is uniquely identified by a number in the header of the packet (see field CID in Table 1). A *source* may output multiple streams data, each for a different universe. Also, multiple *sources* may output data for a given universe.

3.6 Active Data Slots: When translating ANSI E1.11 DMX512-A [DMX] to E1.31, the active data slots are defined as ranging from data slot 1 to the maximum data slot in the most recently received packet with the corresponding START Code. Devices originating E1.31 shall define their active data slots using the *DMP First Property Address* and *DMP Property Count* fields shown in Table 1.

4 Protocol Packet Structure Summary

E1.31 is a protocol that lives within the suite of protocols defined by the ANSI E1.17 [ACN] standard. The ACN standard provides a method for layering protocols and for using a simple repeating message structure throughout. The lowest layer ACN protocol is called the Root Layer Protocol (RLP), which wraps E1.31 as well as other protocols such as Session Data Transport (SDT). It is not necessary to implement or understand these other protocols to use E1.31 to send DMX512 [DMX] data over ACN.

The repeating message structure is called the Protocol Data Unit (PDU) format which is fully defined in “Section 2.2 Protocol Data Units and Blocks – The Standard ACN Message Format” and “Section 2.4 PDU Fields” of the “ACN Architecture Document” of ANSI E1.17 [ACN].

E1.31 defines an outer layer PDU wrapper that specifies the sequence number of a packet and that carries a block of data (all PDUs carry a block of data). This data block contains a nested PDU containing a single message of the Device Management Protocol of ANSI E1.17 [ACN] to carry DMX512 [DMX] data. Each PDU contains a length field which equals the length of the entire PDU including its header and data block information.

Table 1 describes the E1.31 packet format on UDP [UDP].

Octet	Field Size	Field Name	Field Description	Field Contents
Root Layer (See Section 5)				
0, 1	2	Preamble Size	Define RLP Preamble Size.	0x0010
2,3	2	Post-amble Size	RLP Post-amble Size.	0x0000
4-15	12	ACN Packet Identifier	Identifies this packet as E1.17	0x41 0x53 0x43 0x2d 0x45 0x31 0x2e 0x31 0x37 0x00 0x00 0x00
16-17	2	Flags and Length	Protocol flags and length	Low 12 bits = PDU length High 4 bits = 0x7
18-21	4	Vector	Identifies RLP Data as 1.31 Protocol PDU	0x00000004
22-37	16	CID	Sender's CID	Sender's unique ID
E1.31 Framing Layer (See Section 6)				
38-39	2	Flags and Length	Protocol flags and length	Low 12 bits = PDU length High 4 bits = 0x7
40-43	4	Vector	Identifies 1.31 data as DMP Protocol PDU	0x00000002
44-107	64	Source Name	User Assigned Name of Source	UTF-8 [UTF-8] encoded string, null-terminated
108	1	Priority	Data priority if multiple sources	0-200, default of 100
109-110	2	Reserved	Reserved	Transmitter Shall Send 0 Receivers Shall Ignore
111	1	Sequence Number	Sequence Number	To detect duplicate or out of order packets.
112	1	Options	Options Flags	Bit 7 = Preview_Data Bit 6 = Stream_Terminated
113-114	2	Universe	Universe Number	Identifier for a distinct stream of DMX Data
DMP Layer (See Section 7)				
115-116	2	Flags and Length	Protocol flags and length	Low 12 bits = PDU length High 4 bits = 0x7
117	1	Vector	Identifies DMP Set Property Message PDU	0x02
118	1	Address Type & Data Type	Identifies format of address and data	0xa1
119-120	2	First Property Address	Indicates DMX START Code is at DMP address 0	0x0000
121-122	2	Address Increment	Indicates each property is 1 octet	0x0001
123-124	2	Property value count	Indicates 1+ the number of slots in packet	0x0001 -- 0x0201
125-637	1-513	Property values	DMX512-A START Code + data	START Code + Data

Table 1: E1.31 Packet Format

All packet contents shall be transmitted in network byte order (big endian).

5 E1.31 use of the ACN Root Layer Protocol

E1.31 shall use the ACN Root Layer Protocol as defined in the ANSI E1.17 [ACN] “ACN Architecture” document. The fields described here are for E1.31 on UDP [UDP]. Alternative Root Layer Protocol (RLP) content may be defined by further standards in order to transport E1.31 on other protocols.

5.1 Preamble Size

Transmitters shall set the Preamble Size to 0x0010. Receivers of UDP [UDP] based E1.31 shall discard the packet if the Preamble Size is not 0x0010. The preamble contains the preamble size field, the post-amble size field, and the ACN packet identifier and has a length of 0x10 octets.

5.2 Post-amble Size

There is no post-amble for RLP used over UDP [UDP]; therefore the Post-amble Size is 0x0.

Transmitters shall set the Post-amble Size to 0x0000. Receivers of UDP based E1.31 shall discard the packet if the Post-amble Size is not 0x0000.

5.3 ACN Packet Identifier

The ACN Packet Identifier shall contain the following sequence of hexadecimal characters 0x41 0x53 0x43 0x2d 0x45 0x31 0x2e 0x31 0x37 0x00 0x00 0x00.

Receivers shall discard the packet if the ACN Packet Identifier is not valid.

5.4 Flags & Length

The Root Layer's Flags & Length field is a 16-bit field with the PDU length encoded in the low 12 bits and 0x7 in the top 4 bits.

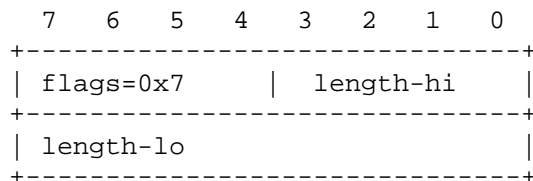


Figure 1: RLP Flags and Length

The RLP PDU length is computed starting with octet 16 and counting all octets in the packet through the last Property Value provided in the DMP layer (Octet 637 for a full payload). This is the length of the RLP PDU.

5.5 Vector

Transmitters shall set the Root Layer's Vector to 0x00000004. Receivers shall discard the packet if the received value is not 0x00000004. This value indicates that the root layer PDU is wrapping an E1.31 framing layer PDU.

5.6 CID (Component Identifier)

The Root Layer contains a CID. The CID shall be a UUID (Universally Unique Identifier) [UUID] that is a 128-bit number that is unique across space and time compliant with RFC 4122 [UUID]. Each piece of equipment should maintain the same CID for its entire lifetime (e.g. by storing it in read-only memory). This means that a particular component on the network can be identified as the same entity from day to day despite network interruptions, power down and so on. However, in some systems there may be situations in which volatile components are dynamically created “on the fly” and in these cases the controlling process can generate CIDs as required. The choice of UUIDs for CIDs allows them to be generated as required without reference to any registration process or authority. As with all E1.31 packet contents, the CID shall be transmitted in network byte order (big endian).

6 E1.31 Framing Layer Protocol

6.1 Flags & Length

The E1.31 Flags & Length field is a 16-bit field with the PDU length encoded in the low 12 bits and 0x7 in the top 4 bits.

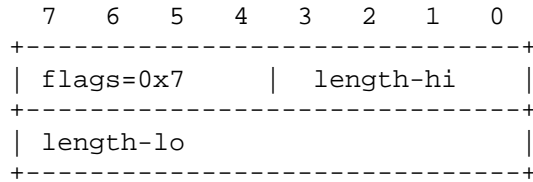


Figure 2: E1.31 Flags and Length

The E1.31 framing layer PDU length is computed starting with octet 38 and continuing through the last property value provided in the DMP PDU (octet 637 for a full payload). This is the length of the E1.31 framing layer PDU.

6.2 Vector

Transmitters shall set the E1.31 Layer's Vector to 0x0000002. Receivers shall discard the packet if the received value is not 0x0000002. This value indicates that the E1.31 framing layer is wrapping a DMP PDU.

6.3 Source Name

A user assigned name provided by the source of the packet for use in displaying the identity of a source to a user. There is no mechanism, other than user configuration of sources, to ensure uniqueness of this name. The source name shall be null terminated. If the source component implements ACN discovery as defined in EPI 19 [ACN] then this name shall be the same as the UACN field specified in EPI 19 [ACN].

6.4 Priority

A receiver conforming to this standard may receive data for the same universe from multiple sources that may be distinguished by examining the CID in the packet. (This is a situation that cannot occur in conventional DMX systems.)

The Priority field is an unsigned one octet field. The value is used by receivers in selecting between multiple sources of data for a given universe number. Sources that do not support variable priority shall transmit a priority of 100. No priority outside the range of 0 to 200 shall be transmitted on the network. Priority increases with numerical value, i.e., 200 is a higher priority than 100.

For a given universe number, an E1.31 receiver shall treat data from packets with the highest priority as the definitive data for that universe.

6.4.1 Multiple Sources at Highest Priority

It is possible for there to be multiple sources, all transmitting data at the highest currently active priority level for a given universe. When this occurs, receivers must handle these sources in some way.

A receiver which is only capable of processing one source of data will encounter a *sources exceeded* condition when two or more sources are present.

Many devices are capable of combining, merging or arbitrating between the candidate sources by some algorithm (see below), but such algorithms frequently limit the number of concurrent sources which can be handled due to resource limitations, or encounter situations where there are still multiple candidate

sources meeting some specified condition, and then, once again, a *sources exceeded* condition arises which requires resolution.

6.4.2 Discussion of Merge and Arbitration Algorithms

A process of combining data from multiple sources to produce a definitive result is called a merge. A process which selects between candidate sources based on some additional selection criterion is called arbitration.

The single most common merging algorithm, which is usually appropriate to lighting intensity data (e.g. dimmer inputs), is to take the highest (numerically largest) level present from any of the candidate sources slot by slot throughout the universe - so-called Highest Takes Precedence (HTP) merging. A variation of this uses DMX512 START Code DDh (see [DMX] and http://www.esta.org/tsp/working_groups/CP/DMXAlternateCodes.php) to indicate slot-by-slot priority before merging the highest priority data for each slot on an HTP basis.

For other devices such as movement axes in automated luminaires, HTP is often highly inappropriate. In this case, it is common to accept only one candidate source, but arbitration criteria may be applied e.g. based on information in the Source Name field.

6.4.3 Discussion of Resolution of Sources Exceeded Condition

Resolution is required when the number of sources exceeds limitations of the algorithm or of resources available. In the simplest case with no merging or arbitration, this occurs when there is more than one source (at highest active priority).

One resolution mechanism is to stop accepting data from any source. Other mechanisms may choose one or more from the candidate sources by some overload selection scheme.

Designers are very strongly discouraged from implementing resolution algorithms that generate different results from the same source combination on different occasions, because this can make *sources exceeded* conditions hard to detect, makes networks very hard to troubleshoot and may cause unexpected results at critical times. For example, an arbitration scheme which accepts the first source detected at the active highest priority and rejects any subsequent ones will produce results dependent on the order in which equipment is initialized and the vagaries of packet timing, and is not recommended.

6.4.4 Requirements for Merging and Arbitrating

The ability of devices to merge or arbitrate between multiple sources at highest active priority shall be declared in user documentation for the device.

If merging or arbitration is implemented, the maximum number of sources which can be correctly handled shall be declared in user documentation for the device.

If merging or arbitration is implemented the algorithm used shall be declared in user documentation for the device.

6.4.5 Requirements for *Sources Exceeded* Resolution

The resolution behavior of equipment under *sources exceeded* conditions shall be declared in user documentation for the device.

Receiving devices conforming are strongly recommended to indicate a *sources exceeded* condition by some means easily detected at the device, e.g., by a flashing indicator, or obvious status message.

Receiving devices may additionally indicate a *sources exceeded* condition by other means such as remote indication initiated by a network message. This is particularly appropriate for devices which may be hard to access.

6.4.6 Requirements for Devices with Multiple Operating Modes

Receiving devices which have multiple configurations available to select between different methods for merging and/or *Sources Exceeded* resolution, shall meet the rules above for each configuration separately. Any configurations in which the device is not compliant with this standard should be clearly declared as such, but are otherwise beyond the scope of this specification.

6.5 Sequence Number

In a routed network environment it is possible for packets to be received in a different order to the one in which they were sent. The sequence number allows receivers or diagnostic equipment to detect out of sequence or lost packets.

Sources shall maintain a sequence for each universe they transmit. The sequence number for a universe shall be incremented by one for every packet sent on that universe.

6.6 Options

This bit-oriented field is used to encode optional flags that control how the packet is used.

Preview_Data: Bit 7 (most significant bit)

This bit, when set to 1, indicates that the data in this packet is intended for use in visualization or media server preview applications and shall not be used to generate live output.

Stream_Terminated: Bit 6

This bit is intended to allow E1.31 transmitters to terminate transmission of a stream without waiting for a timeout to occur and to indicate to receivers that such termination is not a fault condition.

When set to 1, this bit indicates that the source of the data for the universe specified in this packet has terminated transmission of that universe. Three packets containing this bit set to 1 shall be sent by transmitters upon terminating sourcing of a universe. Upon receipt of a packet containing this bit set to a value of 1, a receiver shall enter network data loss condition. Any property values in these packets shall be ignored.

Bits 0 through 5 of this field are reserved for future use and shall be transmitted as 0 and ignored by receivers.

6.7 Universe

The Universe is a 16-bit field that defines the universe number of the data carried in the packet. Universe values shall be limited to the range 1 to 63999. Universe value 0 and those between 64000 and 65535 are reserved for future use. See Section 8 for more information.

6.8 Framing Layer Operation and Timing - Transmitter Requirements

6.8.1 Transmission Rate

E1.31 sources shall not transmit packets for a given universe number at a rate which exceeds the maximum refresh rate specified in E1.11, for a data packet with the same slot count. Note that E1.11 [DMX] places special restrictions on the maximum rate for alternate START Code packets in Section 8.5.3.2 of that document.

6.8.2 Null START Code Transmission Requirements

In order to avoid unnecessary usage of network bandwidth, transmission of redundant Null START Code data is minimized. For a given universe number, transmitting devices shall transmit Null START Code data only when that data changes, with the following exceptions:

1. Three packets containing the non-changing data shall be sent before the initiation of transmission suppression.
2. Thereafter, a single keep-alive packet shall be transmitted at intervals of between 800mS and 1000mS. Each keep-alive packet shall have identical content to the last Null START Code data packet sent with the exception that the sequence number shall continue to increment normally.

These requirements do not apply to alternate START Code data.

6.9 Framing Layer Operation and Timing - Receiver Requirements

6.9.1 Network Data Loss

Network data loss condition is a condition that is defined as either the absence of reception of E1.31 packets from a given source for a period of 2.5 seconds or the receipt of a packet containing the Options field, bit 6 set to value of 1 (see Section 6.6 Options). Data loss is specific to a universe (see the Universe field in Table 1), so data loss may exist for one universe from a source and not for other universes provided by that same source.

When a data loss condition arises, a source specific universe is considered disconnected.

6.9.2 Sequence Numbering

Receivers that do not support sequence numbering of packets shall ignore the contents of this field.

Having first received a packet with sequence number A, a second packet with sequence number B arrives. If, using signed 8-bit binary arithmetic, $B - A$ is less than or equal to 0, but greater than -20 then the packet containing sequence number B shall be deemed out of sequence and discarded.

Note: This algorithm allows the sequence stream from a source to jump by large amounts without undue delay, as in the case of a reset, without allowing packets received slightly out of order to cause flicker or interfere with predictive algorithms found in many moving light fixtures.

For receivers supporting sequence numbering, packets shall be processed in the order received unless they are discarded according to the above algorithm.

7 DMP Layer Protocol

In DMP terms the DMX packet is treated at the DMP layer as a set property message for an array of up to 513 one-octet virtually addressed properties. A restriction of E1.31 is that the array shall always begin at property zero corresponding in DMX nomenclature to START Code. This allows E1.31 implementations which do not process DMP generically to treat much of the DMP header content as fixed values.

7.1 Flags & Length

The DMP Layer's Flags & Length field is a 16-bit field with the PDU length encoded in the low 12 bits and 0x7 in the top 4 bits.

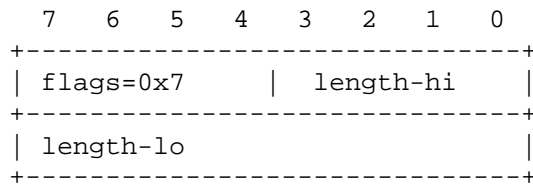


Figure 3: DMP Flags and Length

The DMP layer PDU length is computed starting with octet 115 and continuing through the last property value provided in the DMP PDU (octet 637 for a full payload). This is the length of the DMP PDU.

7.2 Vector

The DMP Layer's Vector shall be set to 0x02, which indicates a DMP Set Property message by transmitters. Receivers shall discard the packet if the received value is not 0x02.

7.3 Address Type and Data Type

Transmitters shall set the DMP Layer's Address Type and Data Type to 0xa1. Receivers shall discard the packet if the received value is not 0xa1.

7.4 First Property Address

Transmitters shall set the DMP Layer's First Property Address to 0x0000. Receivers shall discard the packet if the received value is not 0x0000.

7.5 Address Increment

Transmitters shall set the DMP Layer's Address Increment to 0x0001. Receivers shall discard the packet if the received value is not 0x0001.

7.6 Property Value Count

The DMP Layer's Property Value Count is used to encode the number of DMX512-A [DMX] Slots (including the START Code slot).

7.7 Property Values (DMX512-A Data)

The DMP Layer's Property values field is used to encode the DMX512-A [DMX] START Code and data.

The first octet of the property values field shall be the DMX512-A [DMX] START Code.

The remainder of the property values shall be the DMX512-A data slots. This data shall contain a sequence of octet data values that represent a consecutive group of data slots, starting with slot 1, from a DMX512-A packet. The number of data slots encoded in the frame shall not exceed the DMX512-A limit of 512 data slots.

Processing of Alternate START Code data shall be compliant with ANSI E1.11 [DMX] Section 8.5.3.3 - Handling of Alternate START Code packets by in-line devices.

8 Operation of E1.31 in IPv4 Networks

8.1 Assignment of IP Addresses

E1.31 compliant equipment requiring an IP address shall conform to EPI-29 (Revised Rules for Allocation of Internet Protocol Version 4 Addresses to ACN Hosts) of ANSI E1.17 [ACN] for IP-address assignment.

8.2 Association of Multicast Addresses and Universe

This standard uses network multicast addressing in order to direct DMX512-A universes to their specified destination. Universes may be sent directly between network devices using unicast addressing, however no discovery mechanism is specified in this standard to support the dynamic determination of such addresses.

Addressing and partitioning of multicast traffic is achieved by setting the least significant two bytes of the multicast IP address to the desired universe number. Multicast addresses corresponding to the top 256 universes are reserved by the IANA for scope relative addressing; therefore E1.31 devices shall not transmit on these top 256 addresses. If these reserved universes are used in the future, they must be transmitted unicast. Additionally, E1.31 devices shall not use universe number 0 as it is reserved for future use. Also, Universe numbers between 64000 and 65535 are reserved for future use and shall not be used by E1.31 devices except as specified in future standards designed to coexist with E1.31 and which specifically call for the use of these universes.

Controllers can thus transmit data to the required universes without prior knowledge of the network topology. Equally, responders can listen to a predetermined IP address for data representing a specific universe.

An operating mode shall be provided where E1.31 controllers shall transmit universes on the multicast address as defined in Table 2. E1.31 receivers of a universe shall subscribe to the multicast address defined in Table 2. Note: The identity of the universe shall be determined by the universe number in the packet and not assumed from the multicast address. An E1.31 receiver shall also respond to 1.31 data received on its unicast address.

8.3 Multicast Subscription

Transmitters and Receivers shall use IGMP V2 [IGMP2] or IGMP V3 [IGMP3]. IGMP is used to communicate multicast address usage to network routers.

This requirement is placed on transmitters because many switches or routers do not handle incoming multicast packets well unless the source has advertised in this manner.

8.4 Allocation of Multicast Addresses

Multicast addresses are from the IPv4 Local Scope and will be managed by routers in conformance with RFC 2365 [ASIPM].

The multicast IP address is defined in Table 2 below:

IP Address Byte	Value
1	239
2	255
3	Universe – Hi byte
4	Universe – Lo byte

Table 2: Universe - IP mapping

When multicast addressing is used, the UDP destination Port shall be set to the standard ACN-SDT multicast port (5568). For unicast communication the ACN-SDT multicast port shall be used by default, but methods for configuration and use of alternative ports may be provided.

9 Translation between DMX512-A and E1.31

9.1 DMX512-A to E1.31 Translation

Devices performing translation of incoming DMX512-A [DMX] data to E1.31 network data are subject to the requirements of this Section.

9.1.1 Boot Condition

A DMX512-A [DMX] to E1.31 translator shall not transmit E1.31 packets for a given universe until it has received at least one valid (properly formed) DMX512-A [DMX] input packet for that universe.

9.1.2 Temporal Sequence

A DMX512-A [DMX] to E1.31 translator shall transmit packets in the order in which they are received from the DMX512-A [DMX] source.

9.1.3 Loss of Data

Upon detection of loss of data as defined in DMX512-A [DMX], a transmitter shall terminate transmission in accordance with Section 6.6.

9.2 E1.31 to DMX512-A Translation

9.2.1 General

Devices performing translation of incoming E1.31 network data to DMX512-A [DMX] data are subject to the requirements of this Section.

9.2.2 Loss of Data

An operating mode shall be provided, whereupon detection of loss of data, as defined in Section 6.9.1, for all sources of a universe, a transmitter shall immediately stop transmitting DMX512-A [DMX] packets. In addition, a transmitter may supplement this required mode with alternative operating modes, for example, such as those implementing a hold-last-look feature by continuously retransmitting the last valid packet.