

ANSI/CTA Standard

Serial Communication Protocol for Portable Electronic Devices

ANSI/CTA-2017.1 R-2013

(Formerly ANSI/CEA-2017.1 R-2013)

November 2007



NOTICE

Consumer Technology Association (CTA)™ Standards, Bulletins and other technical publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such Standards, Bulletins and other technical publications shall not in any respect preclude any member or nonmember of the Consumer Technology Association from manufacturing or selling products not conforming to such Standards, Bulletins or other technical publications, nor shall the existence of such Standards, Bulletins and other technical publications preclude their voluntary use by those other than Consumer Technology Association members, whether the standard is to be used either domestically or internationally.

Standards, Bulletins and other technical publications are adopted by the Consumer Technology Association in accordance with the American National Standards Institute (ANSI) patent policy. By such action, the Consumer Technology Association does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the Standard, Bulletin or other technical publication.

This document does not purport to address all safety problems associated with its use or all applicable regulatory requirements. It is the responsibility of the user of this document to establish appropriate safety and health practices and to determine the applicability of regulatory limitations before its use.

This document is copyrighted by the Consumer Technology Association and may not be reproduced, in whole or part, without written permission. Federal copyright law prohibits unauthorized reproduction of this document by any means. Organizations may obtain permission to reproduce a limited number of copies by entering into a license agreement. Requests to reproduce text, data, charts, figures or other material should be made to the Consumer Technology Association.

(Formulated under the cognizance of the CTA **R6 Portable, Handheld and In-Vehicle Electronics Committee.**)

Published by
©CONSUMER TECHNOLOGY ASSOCIATION 2015
Technology & Standards Department
www.CTA.tech

All rights reserved

FOREWORD

This standard was developed by the Consumer Electronics Association's (CEA) Mobile Electronics Committee (R6). It is intended to be a companion document to CEA-2017, Common Interconnection for Portable Media Players.

Copyright © 2013 Consumer Electronics Association

(This page intentionally left blank.)

CONTENTS

1 Scope.....	1
2 References.....	1
2.1 Normative References.....	1
2.1.1 Normative Reference List	1
2.1.2 Normative Reference Acquisition	1
2.2 Definitions	2
2.3 Symbols and Abbreviations	3
2.4 Compliance Notation.....	4
3 Description of Serial Protocol	4
3.1 Serial Protocol Framework	4
3.2 Serial Protocol Logical Structure.....	5
3.2.1 Function Blocks.....	6
3.2.2 Functions.....	6
3.2.3 Function Operations.....	6
3.2.3.1 Function Types.....	7
3.2.3.2 Method Function Type.....	7
3.2.3.3 Property Function Type.....	7
3.2.3.4 Event Function Type.....	8
3.2.4 Data Transport	8
3.2.4.1 Data Transport Packet Format.....	8
3.2.4.2 Data Transport Packet Checksum Computation	8
3.2.5 Data Types.....	8
3.3 Serial Protocol Logical Device Model Communication	10
3.3.1 Serial Protocol Host Device States	10
3.3.1.1 HDEV INOP STATE	10
3.3.1.2 HDEV INIT STATE.....	10
3.3.1.3 HDEV SYNC STATE	10
3.3.1.4 HDEV GET FUNCTION BLOCKS STATE.....	11
3.3.1.5 HDEV OPER STATE	11
3.3.1.6 HDEV Events	11
3.3.1.7 HDEV State Transitions	13
3.3.2 Serial Protocol Portable Electronic Device States	14
3.3.2.1 PEDEV INOP STATE	14
3.3.2.2 PEDEV INIT STATE	14
3.3.2.3 PEDEV SYNC STATE	14
3.3.2.4 PEDEV WAIT FOR FUNCTION BLOCKS REQUEST STATE.....	14
3.3.2.5 PEDEV OPER STATE.....	15
3.3.2.6 PEDEV Events	15
3.3.2.7 PEDEV State Transitions.....	16
3.3.3 Serial Protocol Standardized Times	17
3.3.4 Serial Protocol Start Up and Function Block Notification	18
3.3.5 Serial Protocol Function Notification	21
3.3.6 Serial Protocol Device Information	24
3.3.7 Serial Protocol Remote Control	27
3.3.8 Serial Protocol Device Database Browsing and Playback Control	32
3.3.9 Serial Protocol Small Image Transfer.....	43
3.4 Serial Protocol Defined Function Blocks.....	45
3.4.1 Function Block NetBlock	45
3.4.2 Function Block AudioAmplifier	45
3.4.3 Function Block AuxIn.....	45
3.5 Serial Protocol Function Block Tables.....	46

3.6 Serial Protocol Error Handling	90
--	----

Tables

Table 1: OPTypes for Properties and Methods.....	7
Table 2: AuxIn Function FktIDs (FBlock ID 24h, FktID 000h).....	47
Table 3: AuxIn Function Notification (FBlock ID 24h, FktID 001h).....	48
Table 4: AuxIn Function NotificationCheck (FBlock ID 24h, FktID 002h).....	50
Table 5: AuxIn Function DeviceStatus (FBlock ID 24h, FktID 207h).....	51
Table 6: AuxIn Function TimePosition (FBlock ID 24h, FktID 201h).....	54
Table 7: AuxIn Function ChapterPosition (FBlock ID 24h, FktID 206h).....	58
Table 8: AuxIn Function DeviceBrowsing (FBlock ID 24h, FktID 43Ah).....	60
Table 9: AuxIn Function TrackInformation (FBlock ID 24h, FktID 434h).....	61
Table 10: AuxIn Function AuxTimeInformation (FBlock ID 24h, FktID 438h).....	63
Table 11: AuxIn Function Repeat (FBlock ID 24h, FktID 452h).....	66
Table 12: AuxIn Function Shuffle (FBlock ID 24h, FktID 453h).....	68
Table 13: AuxIn Function SelectListInfo (FBlock ID 24h, FktID 4A5h).....	70
Table 14: AuxIn Function ImageInformation (FBlock ID 24h, FktID 4B0h).....	73
Table 15: AuxIn Function TrackImageInformation (FBlock ID 24h, FktID 4B1h).....	75
Table 16: AuxIn Function TrackImageTime (FBlock ID 24h, FktID 4B2h).....	76
Table 17: AuxIn Function ImageTransfer (FBlock ID 24h, FktID 4B3h).....	78
Table 18: NetBlock Function FBlockIDs (FBlock ID 01h, FktID 000h).....	79
Table 19: NetBlock Function DeviceInfo (FBlock ID 01h, FktID 001h).....	80
Table 20: AudioAmplifier Function FktIDs (FBlock ID 22h, FktID 000h).....	82
Table 21: AudioAmplifier Function Notification (FBlock ID 22h, FktID 001h).....	83
Table 22: AudioAmplifier Function NotificationCheck (FBlock ID 22h, FktID 002h).....	85
Table 23: AudioAmplifier Function Mute (FBlock ID 22h, FktID 113h).....	86
Table 24: AudioAmplifier Function Bass (FBlock ID 22h, FktID 202h).....	87
Table 25: AudioAmplifier Function Treble (FBlock ID 22h, FktID 203h).....	88
Table 26: AudioAmplifier Function Volume (FBlock ID 22h, FktID 400h).....	89
Table 27: Error Codes.....	90

Figures

Figure 1: Framework of CEA-2017 Device.....	5
Figure 2: Host Device State Transitions	13
Figure 3: Portable Electronic Device State Transitions	16
Figure 4: Normal System Start Up.....	18
Figure 5: Retry System Start Up	19
Figure 6: Error System Start Up	20
Figure 7: AuxIn Function Block Function Query	21
Figure 8: AudioAmplifier Function Block Function Query	21
Figure 9: AuxIn Function Block Function Notification Clear.....	22
Figure 10: AudioAmplifier Function Block Function Notification Clear.....	22
Figure 11: AuxIn Function Block Function Notification Set	23
Figure 12: AuxIn Function Block Function Notification Events.....	23
Figure 13: Device Information Get Company Name	24
Figure 14: Device Information Get Product Name	24
Figure 15: Device Information Get Legal Copyright.....	25
Figure 16: Device Information Get Legal Trademarks	25
Figure 17: Device Information Get Device Name.....	26
Figure 18: Remote Control Play	27
Figure 19: Remote Control Pause	27
Figure 20: Remote Control Menu	28
Figure 21: Remote Control Select	28

Figure 22: Remote Control Play Next Track	29
Figure 23: Remote Control Play Previous Track.....	29
Figure 24: Remote Control Play Next Playlist	30
Figure 25: Remote Control Play Previous Playlist	30
Figure 26: Remote Control Volume Up	31
Figure 27: Remote Control Volume Down	31
Figure 28: Database Browsing Select Top Most Menu	32
Figure 29: Database Browsing Get Current Menu Number of Items	32
Figure 30: Database Browsing Get Current Menu List Information	33
Figure 31: Database Browsing Select Menu Item	34
Figure 32: Database Browsing Get Current Menu List Information	35
Figure 33: Get Play Status.....	36
Figure 34: Get Playback Current Track Time Position	36
Figure 35: Playback Fast Forward Track	37
Figure 36: Playback Rewind Track	37
Figure 37: Get Playback Total Current Track Time	38
Figure 38: Get Playback Current Track Position	38
Figure 39: Playback Next Track	39
Figure 40: Playback Previous Track	39
Figure 41: Get Playback Audiobook Chapter Position	40
Figure 42: Playback Next Audiobook Chapter	40
Figure 43: Playback Previous Audiobook Chapter	41
Figure 44: Playback Repeat Tracklist	41
Figure 45: Playback Shuffle Tracklist	42
Figure 46: Get Device Image Formats.....	43
Figure 47: Get Track Image Formats.....	43
Figure 48: Get Image Track Time	44
Figure 49: Image Transfer	44
Figure 50: Error Message Processing Flow Chart.....	92

(This page intentionally left blank.)

Serial Communication Protocol for Portable Electronic Devices

1 Scope

This document describes a serial communication protocol that enables command and control communication between portable electronic devices and accessories attached to those devices. This protocol builds upon functions provided by the MOST[®] network developed by the MOST Cooperation (www.mostcooperation.com). MOST[®] is a registered trademark of SMSC (www.smsc.com).

2 References

2.1 Normative References

The following standards contain provisions that, through reference in this text, constitute normative provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed in section 2.1.1.

2.1.1 Normative Reference List

MOST Specification, Rev.2.5, 10/2006

MOST FunctionBlock AuxIn, Rev. 3.0-01, 7/2006

Universal Serial Bus Specification Revision 1.1, 9/23/1998

2.1.2 Normative Reference Acquisition

MOST Specifications:

- MOST Cooperation, Bannwaldallee 48, D-76185 Karlsruhe, Germany, Phone +49 (0)721 9 66 50 00
Internet: www.mostcooperation.com , Email: contact@mostcooperation.com

USB Specifications:

- USB Implementers Forum, 5440 SW Westgate Drive, Suite 217, Portland, OR 97221; Phone: +1 503 296 9892; Internet: www.usb.org

2.2 Definitions

CEA-2017 Device

A CEA-2017 device is a physical unit that can be connected to another CEA-2017 device through the physical interface defined in CEA-2017, *Common Interconnection for Portable Media Players*.

Function Block

A group of functions for a CEA-2017 device. A single CEA-2017 device may have multiple components and therefore multiple function blocks. For example, a typical portable media player might have a function block for an auxiliary component, and a function block for an audio amplifier.

Function

A component of a function block. For example, a portable media player function block might contain the functions DeviceStatus, and DeviceBrowsing.

Host Device

A device to which a portable electronic device may be connected for the purposes of controlling the portable electronic device and/or displaying or playing content from the portable electronic device.

Portable Electronic Device

An electronic device that may be carried around by its user. It may record and/or play back audio and/or video, and may also receive and send other data.

MOST Device

A MOST device is a physical unit, which can be connected to a MOST network via a MOST network interface controller.

MOST Specifications

The collective group of MOST specifications, including the MOST Specification, the MOST Specification of Physical Layer, the MOST Function Catalog, the MOST FunctionBlock AuxIn, the MOST Specification Framework, the MOST Medium Access Control Mechanism Specification, the MOST High Protocol Specifications, the MOST Content Protection Scheme – DTCP Implementation, and the MOST Content Security Specification.

2.3 Symbols and Abbreviations

CEA	Consumer Electronics Association
MOST	Media Oriented Systems Transport
Host Device	HDEV
Portable Electronic Device	PEDEV
Human Machine Interface	HMI
Identifier	ID
Function Block	FBlock
Function Block ID	FBlockID
Function ID	FktID
Operation Type	OPType
Second	s
Millisecond	ms

2.4 Compliance Notation

As used in this document “shall” and “must” denote mandatory provisions of this standard. “Should” denotes a provision that is recommended but not mandatory. “May” denotes a feature whose presence does not preclude compliance, and implementation of which is optional. “Optional” denotes items that may or may not be present in a compliant device.

3 Description of Serial Protocol

A CEA-2017 device is a physical unit that can be connected to another CEA-2017 device through the physical interface defined in CEA-2017. The serial physical interface defined in CEA-2017 is the serial RX and TX lines of a CEA-2017 device.

The serial protocol is intended to be used within the framework of the serial physical interface (the RX and TX serial data signals) defined in CEA-2017. A CEA-2017 device communicating over the CEA-2017 defined serial link transmits or receives data over this physical interface in the format described in CEA-2017, which is as follows.

The serial data signal shall have the following characteristics:

1. Four baud rates (9600, 19,200, 57,600 and 115,200 bps)
2. 8 data bits, no parity, one stop bit
3. PEDEV shall auto detect the baud rate based on a synchronization byte (AB hex) sent from the HDEV
4. No flow control
5. When the PEDEV recognizes the baud rate it shall respond at the same baud rate with an acknowledgement (54 hex, which is the complement of the AB hex synchronization byte)
6. During synchronization the HDEV must send the synchronization byte ABh until it gets an acknowledgement byte 54h from the PEDEV or **tSync** time has elapsed.

There is no concept of a control channel, streaming data channel, or packet data channel in the serial protocol. The CEA-2017 serial link is not intended for data streaming which needs high bandwidth, and does not support multimedia data transfer, such as audio or video files.

The CEA-2017 serial link is intended for command and control of a PEDEV which requires low bandwidth and short packet length commands. It may also support transport of small graphics art, such as logos, and album art, or small picture formats from the PEDEV.

3.1 Serial Protocol Framework

The frame work that a CEA-2017 device communicating over the CEA-2017 defined serial link is as follows:

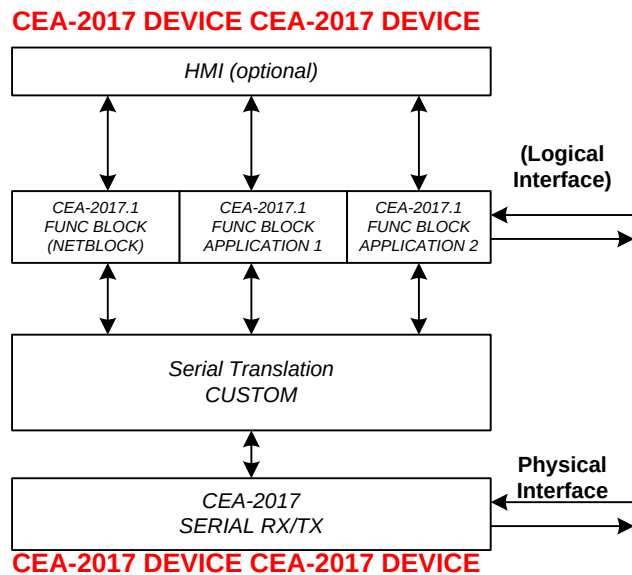


Figure 1: Framework of CEA-2017 Device

The highest part of the frame work may be a HMI. The HMI is completely a custom architecture who's layering and structure will not depend on this serial protocol's standard. The HMI allows a user interface which initiates the serial protocol.

The next layer is the serial protocol's standard function blocks. The serial protocol functions for a PEDEV/HDEV are grouped together in function blocks.

In addition to the function blocks which represent applications, each serial protocol implementation must have a special function block called the NetBlock. The NetBlock provides functions related to the entire device.

Interaction with a function block requires two partners which are referred to as the "PEDEV" and the "HDEV".

The command operational types of the function reside in the HDEV and the report operational types of the function reside in the PEDEV. The HDEV sends commands to a PEDEV and in return receives reports or status from the PEDEV.

A PEDEV must be associated with only one HDEV, and vice versa. The serial protocol defined in this standard only supports point to point communication typically between a HDEV and PEDEV.

3.2 Serial Protocol Logical Structure

This section gives a description of the logical structure of the serial protocol.

In order to distinguish between the different function blocks and functions of a PEDEV or HDEV, each function and function block has an identifier.

FBlockID.FktID

3.2.1 Function Blocks

FBlockID identifies a function block. A function block is a group of functions for a device. A single device may have multiple components and therefore multiple function blocks. FBlockID consists of 8 bits, and has a limit of 255 maximum number of function blocks.

3.2.2 Functions

FktID identifies a specific function within a device that can be called via the serial protocol. FktID consists of 12 bits, but each function block has a limit of 508 maximum number of functions. The range of FktIDs is subdivided into groups that have specific purposes. The subdivision of FktIDs is as follows:

- General Functions (000h...9FFh)
 - General functions in an function block
- Player Specific Proprietary Functions (A00h...FFEh)
 - Proprietary functions that the portable electronic device manufacturer may use
- Accessory Specific Proprietary Functions (A00h...FFEh)
 - Proprietary functions that the portable electronic device accessory manufacturer may use
- Proprietary/Supplier Specific (F00h...FFEh)
 - Functions that may be used by suppliers for any proprietary purpose. Supplier specific functions are not reported in FBlockID.FktIDs.Status. If a supplier specific property supports notification, notification of this property is not affected by the FBlockID.Notification.Set (SetAll) command. If a supplier specific property supports notification, notification of this property is cleared by the FBlockID.Notification.Set (ClearAll) command.

3.2.3 Function Operations

When accessing functions, certain operations are applied to the respective function which is either a property of the function block or a method it provides. The type of operation is specified by the OPType. The parameters of the operation follow the OPType, resulting in the following structure:

FBlockID.FktID.OPType(Data)

OPType identifies the operation that shall be applied to the property or method specified in the FktID.

A property is a segment of computer code that may be used to read and write data, and is frequently used to set or get method calls.

A method is a segment of computer code that performs an action, and may include a set of input parameters that customizes the action and an output value.

The meaning of the OPType value is different depending on whether FktID is a property or a method. OPType consists of 4 bits, so each function may have up to 16 OPType values. The possible meanings for the various OPType values are listed in Table 1.

Table 1: OPTypes for Properties and Methods

OPType	For Properties	For Methods
Commands:		
0	Set	Start
1	Get	Abort
2	SetGet	StartResult
3	Increment	Reserved
4	Decrement	Reserved
5	GetInterface	GetInterface
6	Not allowed	StartResultAck
7	Not allowed	AbortAck
8	Not allowed	StartAck
Reports:		
9	Reserved	ErrorAck
A	Not allowed	ProcessingAck
B	Reserved	Processing
C	Status	Result
D	Not allowed	ResultAck
E	Interface	Interface
F	Error	Error

3.2.3.1 Function Types

A function is a defined attribute of a function block through which it communicates with the external world. Functions are subdivided into three classes, “methods”, “properties”, and “events”.

3.2.3.2 Method Function Type

Functions that can be started and which lead to a result after a certain period of time, are called “methods”.

Methods can be used to control function blocks. In general, a method is triggered only once at a certain point of time, for example, starting the auto-scanning of a tuner. Methods can be defined without parameters or with certain parameters that specify their behavior.

After the reception of a method called by a function block, the respective process must be started. If this is not possible, the function block has to return the respective error message to the sender of the method call. This may happen if the addressed function block has no method of that kind, if a wrong parameter was found, or if the current status of the function block prevents execution of the method.

After finishing the process, the controlled function block should report execution to the HDEV. This report may contain results of the process. If a process runs for a long time, it may be useful to return intermediate results before finishing, such as informing the HDEV about the successful start of the process.

3.2.3.3 Property Function Type

Functions for determining or changing the status of a device, which refer to the current properties of a device are called “properties”.

Properties can be read (e.g., device name), written (e.g., device state), or read and written (e.g., current track). For each property the allowed operations are specified.

Within a function block, a property is normally represented by a variable that represents something such as a limit or a status.

3.2.3.4 Event Function Type

Functions for notifying that an event has occurred are called “events” or “notifications”.

3.2.4 Data Transport

Data may be transported between the two devices. The length of the data field is specified by Length. The data of the operation follows the Length, resulting in the following structure:

FBlockID.FktID.OPType.Length(Data)

Length specifies the length of the data field in bytes. It is encoded in 16 bits.

Length = 0x0000 Data field of length 0

Length = 0x0001 Data field of length 1 byte

Length = 0x03F9 Data field of length 1017 bytes

3.2.4.1 Data Transport Packet Format

- Sync
 - 1 Byte
 - HDEV = ABh
 - PEDEV = 54h
- FBlockID
 - 1 Byte
- FktID/OPType
 - 2 Bytes (FktID = 12 most significant bits, OPType = 4 least significant bits)
- Length
 - 2 Bytes
- Data
 - 0 to 1017 Bytes
- Checksum
 - 1 Byte

3.2.4.2 Data Transport Packet Checksum Computation

1. Add all bytes together excluding the synchronization byte
2. Drop all high bits leaving only the lowest 8 bits
3. Get the two's complement of the lowest 8 bits

Step 3 provides the checksum byte.

3.2.5 Data Types

In a data transmission on the serial link, the maximum data field length is 1017 bytes.

Within a data field, none, one, or multiple parameters in any combination of the following data types can be transported. They are transported MSB first. The sign is encoded in the most significant bit and 2's complement coding is used for signed values. There are the following basic data types:

- Boolean
 - 1 byte
 - Only one bit of the byte can be used
- Unsigned Byte
 - 1 byte
- Unsigned Word
 - 2 bytes
- BitField
 - 1, 2, or 4 bytes
 - Size byte = (Mask.Data)
 - Data: $\frac{1}{2}$ Size byte (data content area)
 - Mask: $\frac{1}{2}$ Size byte (masking area)
 - "Mask" is a masking bit field of the same size as the Data Content Area "Data". It indicates to which bits in the Data Content Area of the BitField an operation shall be applied. The LSB of "Mask" masks the LSB of the Data Content:
 - bit k (Mask) = 1 -> apply Operation to bit k (Data)
 - bit k (Mask) = 0 -> do not apply operation to bit k (Data)
- Signed Byte
 - 1 byte
- Signed Word
 - 2 bytes
- Unsigned Long
 - 4 bytes
- Enum
 - 1 byte
- Stream
 - data structure defined per function
- Signed Long
 - 4 bytes
- String
 - variable length = (Identifier.Content.Terminator)
 - MSB first, high byte first notation for strings
 - UTF8 Unicode only supported
 - Identifier
 - starts with 1 byte Identifier 02h
 - Content
 - characters
 - Terminator
 - ends with 1 character null terminator 00h
- Short Stream
 - Variable length = (Length.Content)
 - Length: 1 byte
 - Content: 0 to 255 bytes (length of the stream is maximum 255 bytes)

The current version of the serial protocol uses only strings with the UTF8 character set.

A string is a variable length element that consists of a 1 byte identifier, the actual content and a null terminator. The identifier indicates the character set. Only identifier 02h (Unicode, UTF8 character set) is currently valid for this protocol. All other values are reserved for future use.

Strings are sent MSB-first and high-byte first, starting with an initial identifier (one byte), then the string characters, followed by a null termination. The initial identifier indicates the coding format. Currently the only required string format is UTF8, which has a string ID of 0x02. In this document the Length field for strings indicates the number of string characters and does not include the initial string ID or the null termination. Therefore an empty string in UTF8 (length = 0) is coded as: 0x02, 0x00.

An "empty" string is represented as: 02h, 00h. This is the identifier for UTF8 followed by the null terminator.

3.3 Serial Protocol Logical Device Model Communication

The function block functionality resides in the HDEV and the PEDEV. The HDEV sends commands to a PEDEV and in return receives reports from the PEDEV. The PEDEV executes the commands issued by a HDEV and sends status reports to the HDEV.

3.3.1 Serial Protocol Host Device States

The HDEV states defined below are only applicable to the CEA-2017.1 Serial Protocol application and don't refer to or imply the state of the CEA-2017 device or any other application running on or associated with the CEA-2017 device.

3.3.1.1 HDEV INOP STATE

In the HDEV INOP STATE, the HDEV is waiting for a PEDEV CONN EVENT. Once a PEDEV CONN EVENT (see definition of event in section 3.3.1.6) occurs the state must transition to the HDEV INIT STATE.

3.3.1.2 HDEV INIT STATE

In the HDEV INIT STATE, the HDEV does a custom initialization which brings its state to that in which the CEA-2017.1 Serial Protocol must start functioning normally, and be able to communicate with the connected PEDEV over the serial link physical interface.

Once a HDEV INIT READY EVENT (see definition of event in section 3.3.1.6) occurs the state must transition to the HDEV SYNC STATE.

If a PEDEV DISCONN EVENT (see definition of event in section 3.3.1.6) occurs the state must transition to the HDEV INOP STATE.

3.3.1.3 HDEV SYNC STATE

In the HDEV SYNC STATE, the HDEV must send the synchronization byte ABh continuously for up to **tSync** standardized time to the PEDEV until the PEDEV acknowledges by returning the synchronization byte 54h.

If the PEDEV does not acknowledge the synchronization byte from the HDEV within **tSync** standardized time of when it was first sent, which is the TIMEOUT RECEIVE ACK SYNC EVENT, then the HDEV must transition to the HDEV INOP STATE.

Once a RECEIVE ACK SYNC EVENT (see definition of event in section 3.3.1.6) occurs the state must transition to the HDEV GET FUNCTION BLOCKS STATE.

If a PEDEV DISCONN EVENT (see definition of event in section 3.3.1.6) occurs the state must transition to the HDEV INOP STATE.

3.3.1.4 HDEV GET FUNCTION BLOCKS STATE

In the HDEV GET FUNCTION BLOCKS STATE, the HDEV must query the PEDEV for the function blocks that the PEDEV supports. If the PEDEV does not respond back with a valid return of the function blocks it supports within the standardized time and the 1st TIMEOUT RECEIVE FBLOCKSIDS EVENT (see definition of event in section 3.3.1.6) occurs then the HDEV must retry the query one time. If the PEDEV does not respond back with a valid return of the function blocks it supports within the standardized time on the retry and the 2nd TIMEOUT RECEIVE FBLOCKSIDS EVENT (see definition of event in section 3.3.1.6) occurs then the state must transition to the HDEV INOP STATE.

Once the HDEV receives a valid return of the function blocks that the PEDEV supports from the PEDEV (RECEIVE FBLOCKSIDS EVENT) then the state must transition to the HDEV OPER STATE.

If a PEDEV DISCONN EVENT (see definition of event in section 3.3.1.6) occurs the state must transition to the HDEV INOP STATE.

3.3.1.5 HDEV OPER STATE

In the HDEV OPER STATE, the HDEV may send various command, control, or query messages to the PEDEV via the function blocks and functions that the PEDEV supports. If the PEDEV does not respond back with a valid return of the request within the standardized time and the 1st TIMEOUT RECEIVE RESPONSE EVENT (see definition of event in section 3.3.1.6) occurs then the HDEV must retry the message one time. If the PEDEV does not respond back with a valid return of the request within the standardized time on the retry and the 2nd TIMEOUT RECEIVE RESPONSE EVENT (see definition of event in section 3.3.1.6) occurs then the state must transition to the HDEV INOP STATE.

If a PEDEV DISCONN EVENT (see definition of event in section 3.3.1.6) occurs the state must transition to the HDEV INOP STATE.

3.3.1.6 HDEV Events

The HDEV events which usually cause HDEV state transitions are described below. How each device individually implements the notification of these events to themselves is custom to the individual device.

- PEDEV CONN EVENT
 - Detection of Accessory Power Output +3.3 V as described in CEA-2017
- PEDEV DISCONN EVENT
 - Unavailability of Accessory Power Output +3.3 V as described in CEA-2017
- HDEV INIT READY EVENT
 - Custom initialization of each individual HDEV which must prepare HDEV to be ready to communicate with PEDEV on the CEA-2017 serial link, and prepare HDEV to process the 2017.1 serial protocol normally.
 - HDEV standardized time limit from receiving PEDEV CONN EVENT to HDEV INIT READY EVENT is **tInit** standardized time
- RECEIVE ACK SYNC EVENT
 - HDEV receives the acknowledgement synchronization byte from PEDEV.
- TIMEOUT RECEIVE ACK SYNC EVENT
 - HDEV does not receive the acknowledgement synchronization byte from PEDEV within **tSync** standardized time from HDEV sending first synchronization byte to PEDEV.
- RECEIVE FBLOCKIDS EVENT

- HDEV receives the function blocks within **tRxFBlockIDs** standardized time after HDEV requests PEDEV to send function blocks that the PEDEV supports.
- 1ST TIMEOUT RECEIVE FBLOCKIDS EVENT
 - HDEV does not receive the function blocks that the PEDEV supports from the PEDEV within **tRxFBlockIDs** standardized time after HDEV requested them the first time.
- 2nd TIMEOUT RECEIVE FBLOCKIDS EVENT
 - HDEV does not receive the function blocks that the PEDEV supports from the PEDEV within **tRxFBlockIDs** standardized time after HDEV requested them the second time.
- RECEIVE RESPONSE EVENT
 - After HDEV sends a valid command, or control message to PEDEV that PEDEV supports and HDEV does receive a response within **tRxResponse** standardized time from request.
- 1ST TIMEOUT RECEIVE RESPONSE EVENT
 - After HDEV sends a valid command, or control message to PEDEV that PEDEV supports, the first time, and HDEV does not receive a response within **tRxResponse** standardized time from request.
- 2nd TIMEOUT RECEIVE RESPONSE EVENT
 - After HDEV sends a valid command, or control message to PEDEV that PEDEV supports, the second time, and HDEV does not receive a response within **tRxResponse** standardized time from request.

3.3.1.7 HDEV State Transitions

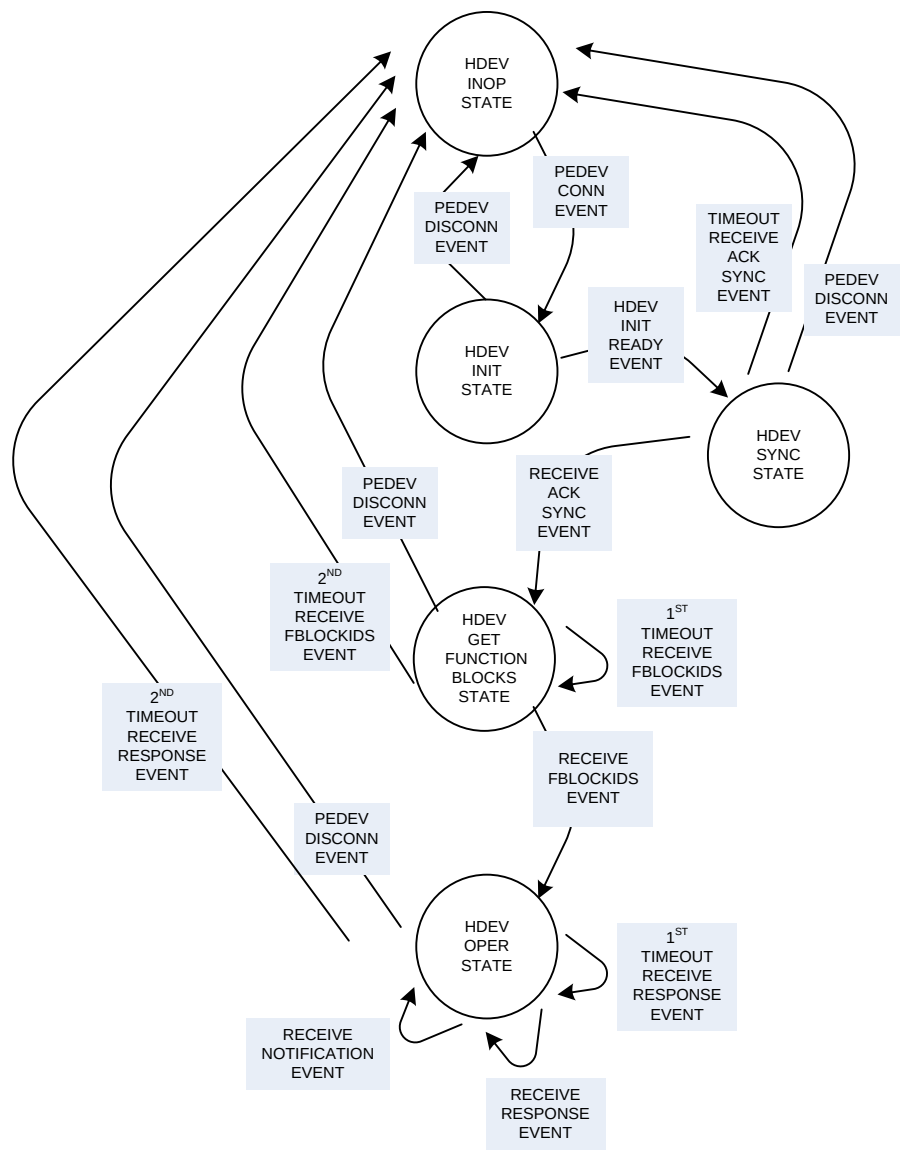


Figure 2: Host Device State Transitions

3.3.2 Serial Protocol Portable Electronic Device States

The HDEV states defined below are only applicable to the CEA-2017.1 Serial Protocol application and don't refer to or imply the state of the CEA-2017 device or any other application running on or associated with the CEA-2017 device.

3.3.2.1 PEDEV INOP STATE

In the PEDEV INOP STATE, the PEDEV is waiting for a HDEV CONN EVENT. Once a HDEV CONN EVENT (see definition of event in section 3.3.2.6) occurs the state must transition to the PEDEV INIT STATE.

3.3.2.2 PEDEV INIT STATE

In the PEDEV INIT STATE, the PEDEV does a custom initialization which brings its state to that in which the 2017.1 serial protocol must start functioning normally, and be able to communicate with the connected HDEV over the serial link physical interface.

Once a PEDEV INIT READY EVENT (see definition of event in section 3.3.2.6) occurs the state must transition to the PEDEV SYNC STATE.

If a HDEV DISCONN EVENT (see definition of event in section 3.3.2.6) occurs the state must transition to the PEDEV INOP STATE.

3.3.2.3 PEDEV SYNC STATE

In the PEDEV SYNC STATE, the HDEV sends the synchronization byte ABh continuously for up to five seconds to the PEDEV until the PEDEV acknowledges by returning the synchronization byte 54h. If the PEDEV does not respond then the HDEV will assume that the PEDEV does not support the serial link communication.

Once the PEDEV does acknowledge by returning the synchronization byte (PEDEV SENT ACK SYNC EVENT (see definition of event in section 3.3.1.6)) then the state must transition to the PEDEV WAIT FOR FUNCTION BLOCKS REQUEST STATE.

If a HDEV DISCONN EVENT (see definition of event in section 3.3.2.6) occurs the state must transition to the PEDEV INOP STATE.

3.3.2.4 PEDEV WAIT FOR FUNCTION BLOCKS REQUEST STATE

In the PEDEV WAIT FOR FUNCTION BLOCKS REQUEST STATE, the PEDEV must wait for the HDEV to send a request for the function blocks that the PEDEV supports. If the HDEV does not send the request within the standardized time and the TIMEOUT REQUEST FBLOCKIDS EVENT (see definition of event in section 3.3.2.6) occurs then the state must transition to the PEDEV INOP STATE.

If the PEDEV does receive the request for the function blocks (REQUEST FBLOCKIDS EVENT), then it must respond back with a valid return of the function blocks it supports within the standardized time (PEDEV SENT FBLOCKIDS EVENT), and the state must transition to the PEDEV OPER STATE.

If a HDEV DISCONN EVENT (see definition of event in section 3.3.2.6) occurs the state must transition to the PEDEV INOP STATE.

3.3.2.5 PEDEV OPER STATE

In the PEDEV OPER STATE, the HDEV may send various command, control, or query messages to the PEDEV via the function blocks and functions that the PEDEV supports. The PEDEV must respond back with a valid response within the standardized time, and the state must stay in the PEDEV OPER STATE.

If a HDEV DISCONN EVENT (see definition of event in section 3.3.2.6) occurs the state must transition to the PEDEV INOP STATE.

3.3.2.6 PEDEV Events

The PEDEV events which usually cause PEDEV state transitions are described below. How each device individually implements the notification of these events to themselves is custom to the individual device.

- HDEV CONN EVENT
 - Refer to CEA-2017 for defined connection requirements.
- HDEV DISCONN EVENT
 - Refer to CEA-2017 for defined disconnect requirements.
- PEDEV INIT READY EVENT
 - Custom initialization of each individual PEDEV which must prepare PEDEV to be ready to communicate with HDEV on the CEA-2017 serial link, and prepare PEDEV to process the 2017.1 serial protocol normally.
 - PEDEV standardized time limit from receiving HDEV CONN EVENT to PEDEV INIT READY EVENT is **tInit**.
- PEDEV SENT ACK SYNC EVENT
 - When PEDEV sends a synchronization byte to HDEV to acknowledge the synchronization byte originally sent from HDEV.
- REQUEST FBLOCKIDS EVENT
 - PEDEV receives a request from HDEV for the function blocks that the PEDEV supports.
- TIMEOUT REQUEST FBLOCKIDS EVENT
 - From the time the PEDEV INIT READY EVENT occurs, the PEDEV waits for the HDEV to send a request for the function blocks that the PEDEV supports and the HDEV does not send the request within the standardized time **tRxFBlockIDReq**.
- PEDEV SENT FBLOCKIDS EVENT
 - When the PEDEV sends the requested for function blocks to the HDEV.
- REQUEST EVENT
 - HDEV sends a command to the PEDEV.

3.3.2.7 PEDEV State Transitions

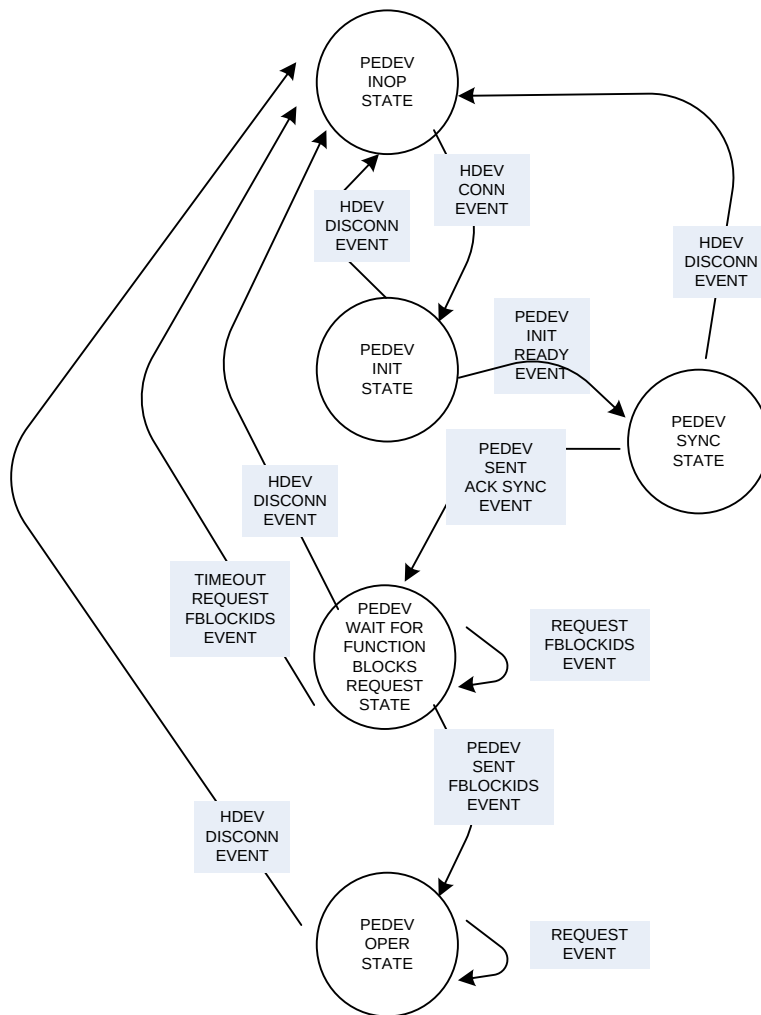


Figure 3: Portable Electronic Device State Transitions

3.3.3 Serial Protocol Standardized Times

The following are the standardized times that must be used for the serial protocol implementation.

tInit	200 ms
tSync	5 s
tRxFBlockIDs	200 ms
tRxResponse	100 ms
tRxFBlockIDReq	5 s

Copyright Consumer Technology Association
Provided by IHS Markit under license with CTA
No reproduction or networking permitted without license from IHS
Full document details: CTA CTA-2017.1 2013

3.3.4 Serial Protocol Start Up and Function Block Notification

During system start up, the PEDEV and HDEV must provide a device connect and an accessory connect signal, respectively, as standardized in CEA-2017, to indicate the physical connection. The HDEV must query for and get the function blocks implemented on the PEDEV.

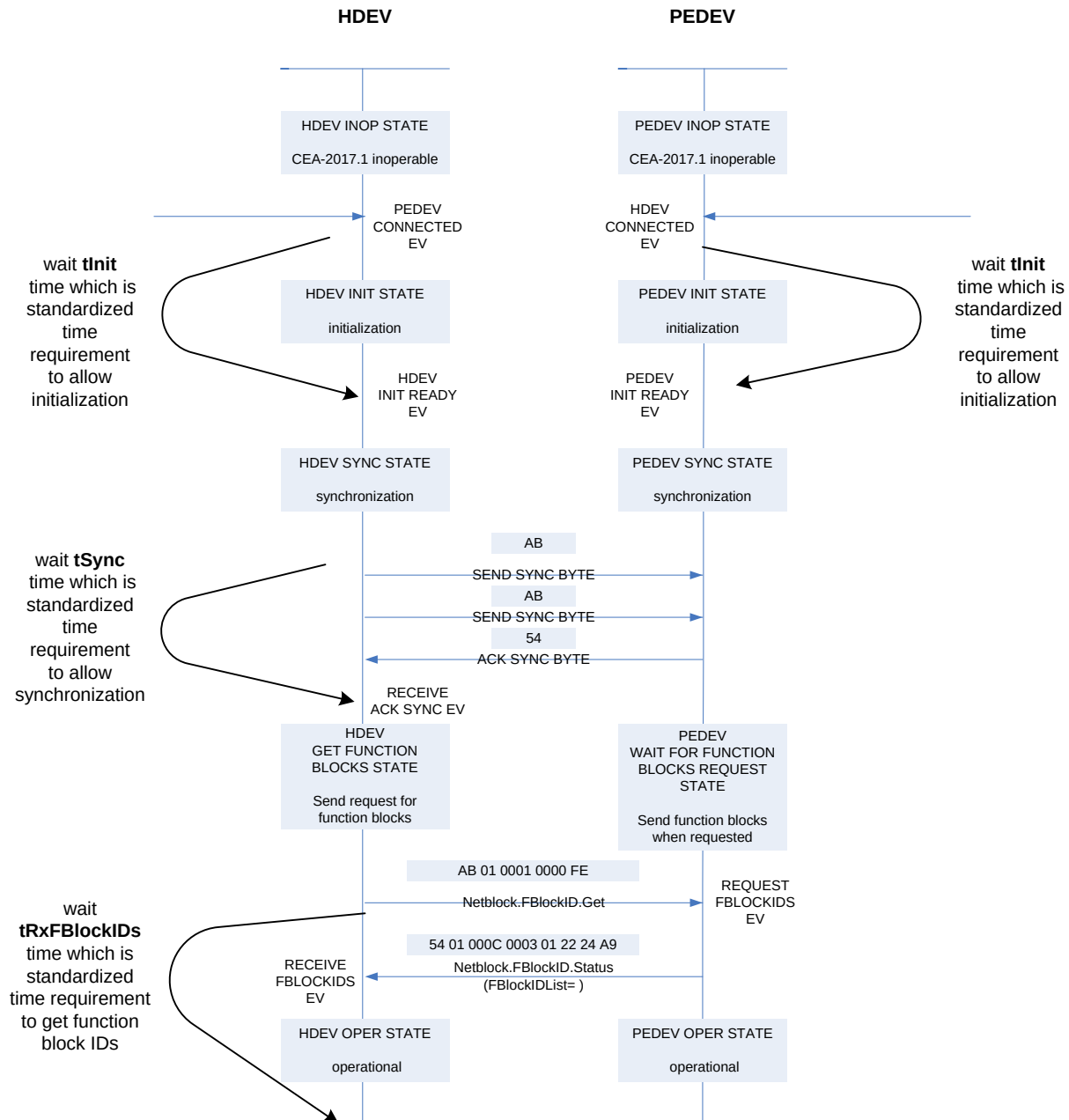


Figure 4: Normal System Start Up

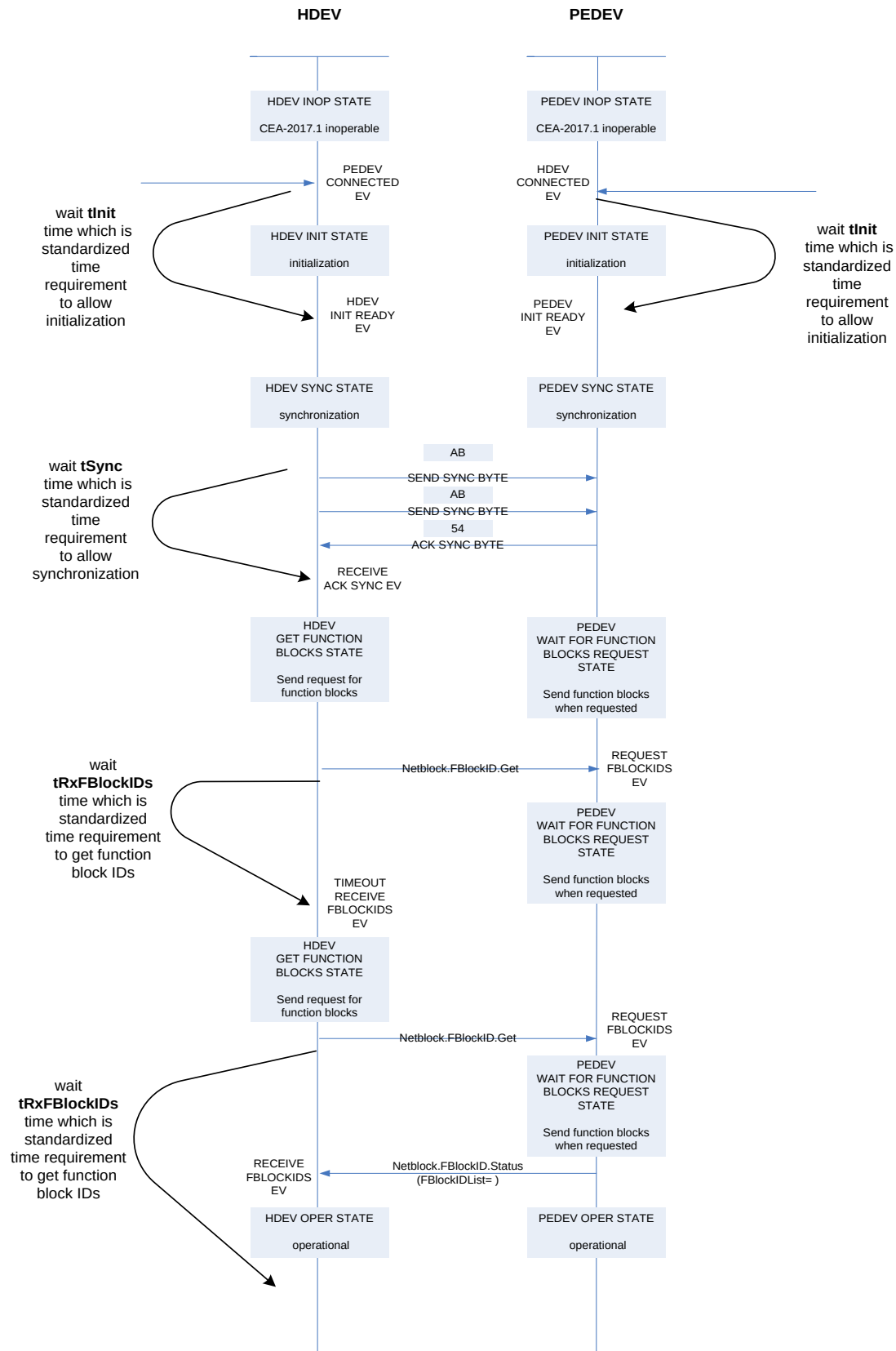


Figure 5: Retry System Start Up

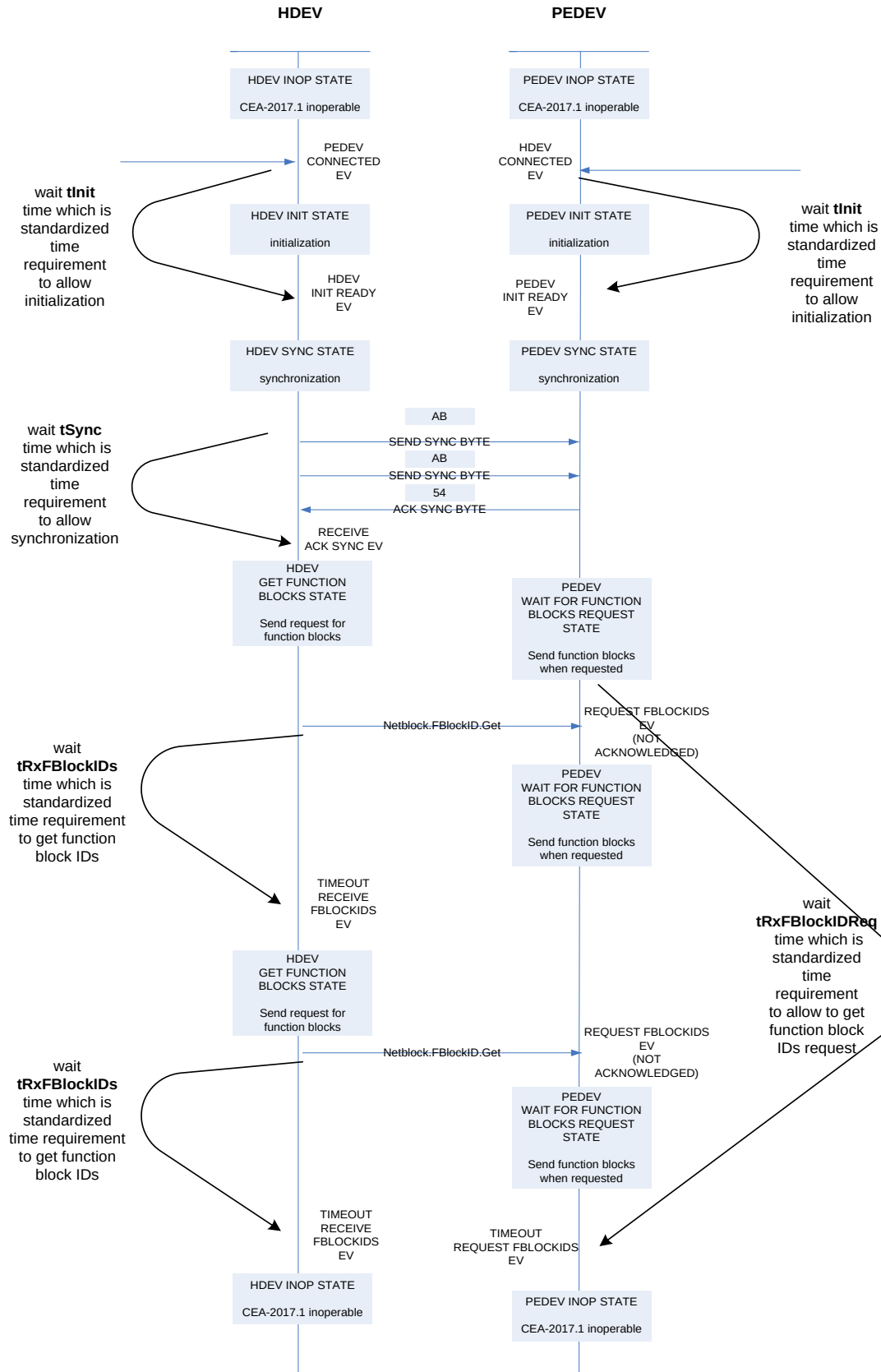


Figure 6: Error System Start Up

3.3.5 Serial Protocol Function Notification

During function notification, the HDEV must query for and get the functions implemented for each function block implemented on the PEDEV. The HDEV must first clear and then may set the function notification matrix for each function block.

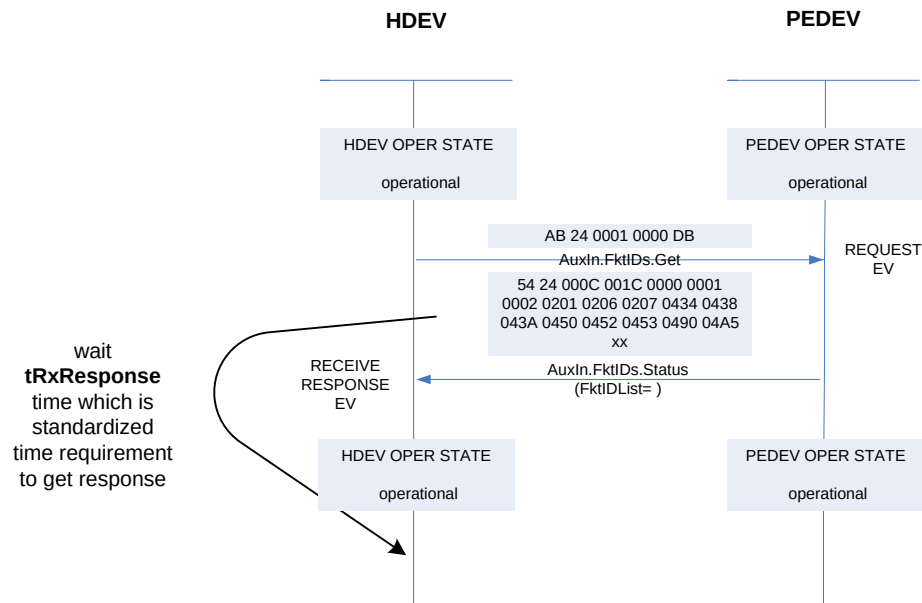


Figure 7: AuxIn Function Block Function Query

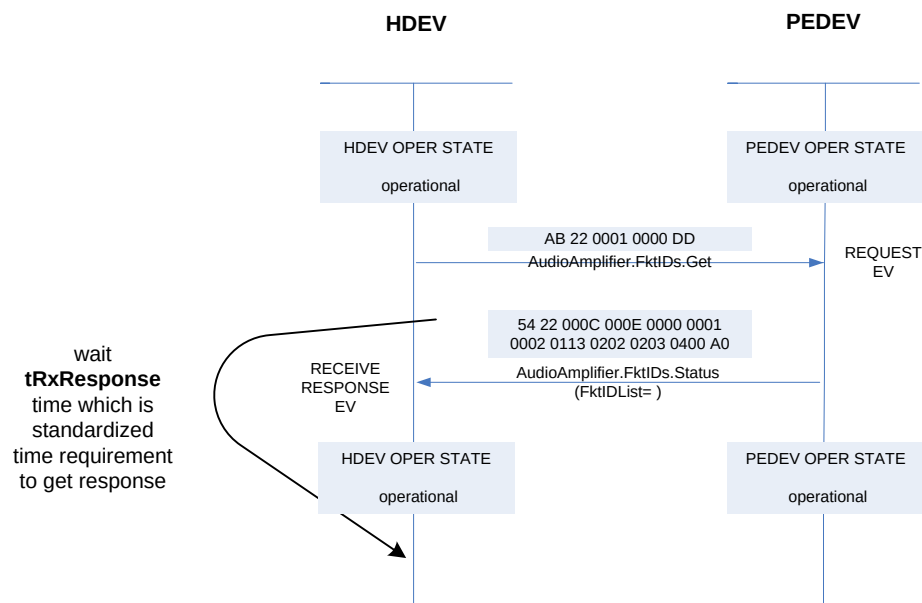


Figure 8: AudioAmplifier Function Block Function Query

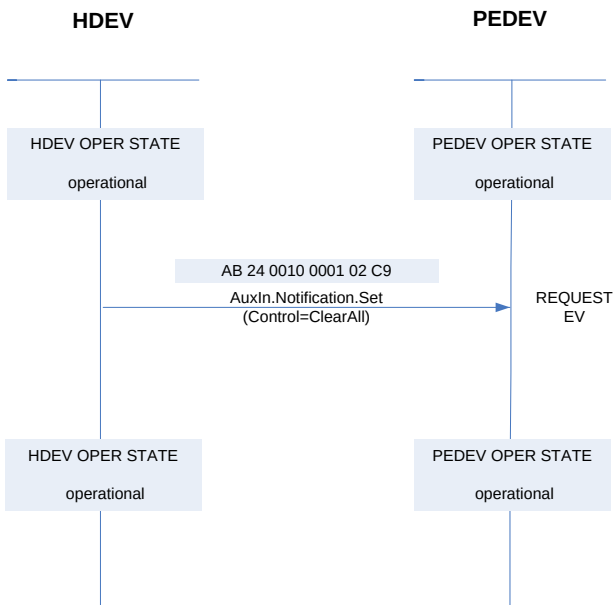


Figure 9: AuxIn Function Block Function Notification Clear

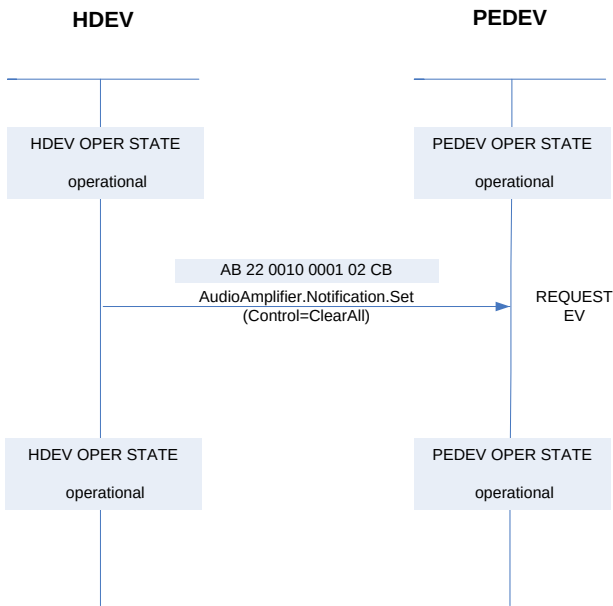


Figure 10: AudioAmplifier Function Block Function Notification Clear

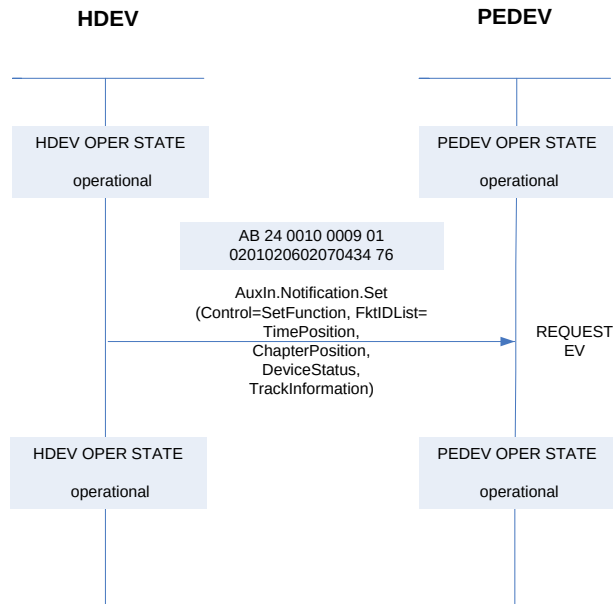


Figure 11: AuxIn Function Block Function Notification Set

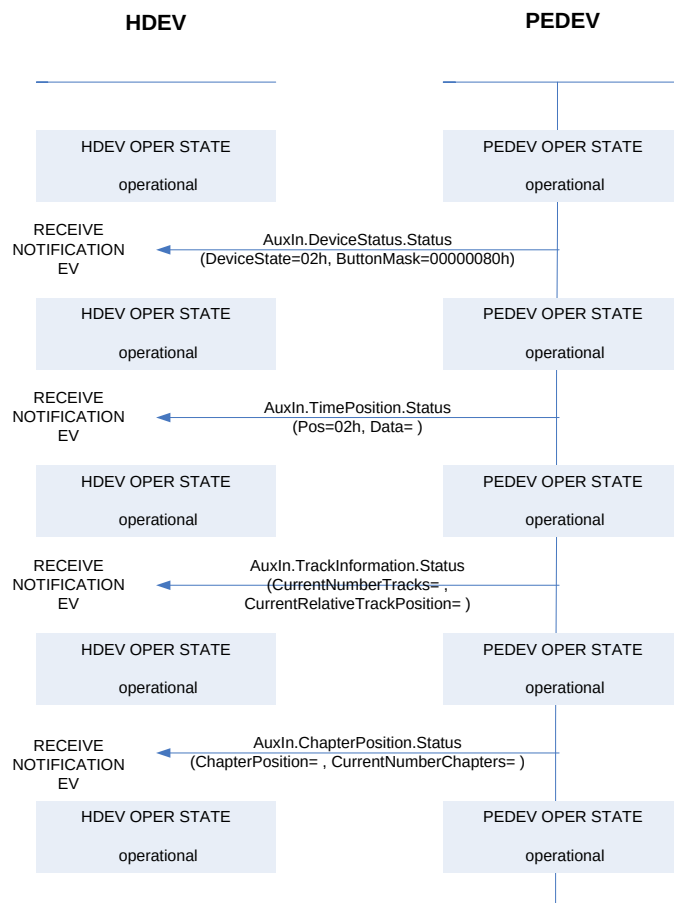


Figure 12: AuxIn Function Block Function Notification Events