Lamb Park, Drewsteignton, Nr Exeter, Devon. UK EX6 6PN

JAMES EMBEDDED SYSTEMS ENGINEERING

DMX512 & RDM E1.20 USB Interface

Programmer's Guide
&
User Interface Specification

For

DMX-TRI MK1 RDM-TRI MK1

Valid From Firmware Revision 00.04.0000

Document Revision N° 40

Lamb Park, Drewsteignton, Nr Exeter, Devon. UK EX6 6PN

JAMES EMBEDDED SYSTEMS ENGINEERING

Notes

Lamb Park, Drewsteignton, Nr Exeter, Devon. UK EX6 6PN

JAMES EMBEDDED SYSTEMS ENGINEERING

Change Record:

Ver	Date	Notes	Rev
01	18 JUL 2009	Document release	17
02	02 FEB 2010	Addition of two Commands and correction of CI_FW_VerGet	30
03	25 FEB 2010	Addition of two Commands, corrections to CI Summary table and SI notation	33
04	21 APR 2010	Updated to include RDM-TRI model and additional ML commands	37
05	16 JUN 2010	Corrections of typographical errors	40

Lamb Park, Drewsteignton, Nr Exeter, Devon. UK EX6 6PN

JAMES EMBEDDED SYSTEMS ENGINEERING

Table of Contents

Lamb Park, Drewsteignton, Nr Exeter, Devon. UK EX6 6PN

1. Overview

1.1 Audience

The guide is intended for developers of applications to target the DMX Transmit-Receive Interface (DMX-TRI) and the Remote Device Management Transmit-Receive Interface (RDM-TRI).

1.2 Purpose

The guide is published to enable developers of hardware and software to make their designs operable with the Transmit-Receive Interfaces (TRI).

1.3 Method

This document specifies the structure of the product command interface to the PC. Command sets are defined and each command is elaborated. Elaborated commands are annotated to assist developers with the intended use for each command, drawing references to the Remote Device Management protocol (RDM) as set out in the standard E1.20-2006.

1.4 References

Where this document refers to RDM ***parameter data,*** details of the associated data structures and application will be found in the E1.20-2006 standard.

2. Device Description

2.1 Hardware

The Interface electronics are manufactured to RHOS standards and mounted in an anodised Aluminium enclosure with black plastic bezels front and back. There should be no requirement to access the electronics within the enclosure. The DMX connection is electrically isolated from the USB port, rated to 1KV.

2.2 Power

The Interfaces are powered from the USB connection to the Host PC and draw a nominal 130mA.

2.3 Connections

Connection to the Host PC is by a standard USB cable, supplied with the unit.
The RDM-TRI is available only with a five pin female XLR DMX connector.
The DMX-TRI is available with three or five pin female XLR connectors fitted to suit the end users' preference.

2.4 Indication

Two mimics on the front panel indicate transmitted and received traffic over the DMX chain. The interface has a sounder to warn of errors when using Legacy commands, in lieu of any provision by the Legacy command set. The sounder is also used to fulfil the requirement of the interface to identify itself when acting as an RDM responder.

2.5 Versions

The Interfaces are available in two versions to accommodate varying requirements in the market place.

2.5.1 **DMX-TRI**

The DMX-TRI uses the advanced technology featured in the RDM-TRI for DMX512 Input and Output functions. The RDM-TRI is RDM protocol aware and is a discoverable RDM responder.

The RDM-TRI may be licensed to be operable with RDM Controller applications available from JESE. The license upgrade path for the DMX-TRI provides cost and environmental benefits, making the device future proof and ready for the proliferation of the RDM E1.20 standard.

2.5.2 **RDM-TRI**

The RDM-TRI supports all commands documented in this document and is suitable for DMX512 Input and Output as well as RDM controller function.

2.6 Protocols

The interfaces provide a rich set of commands for interacting with DMX512 and RDM enabled devices, taking the hard work of learning the complexities of the ANSI E1.20 – 2006 standard.

The interfaces support two command sets, Legacy and Standard commands.

2.6.1 **Legacy**

The Legacy command set makes the unit interoperable with alternative equipment on the market to provide backward compatibility. The legacy commands are very simple to use with a loose binding to the TRI but without the high integrity error handling facilities provided by the standard command set. Errors detected in the Legacy command set are indicated by 'beeps', from an internal sounder. It is recommended that developers of applications utilise the standard Protocol for new applications wherever possible.

2.6.2 **Standard**

The standard command set forms the core of this document and is Ideal for developers who wish to trap and handle errors from remote devices, the DMX connection, the USB connection and violations of the standard in a uniform and seamless way.
All Commands to a TRI solicit an equivalent response to maintain the integrity of the host application

2.7 System

The TRI firmware is an object-based development with interleaved tasks to handle interaction between the Host application and the DMX interface. State machine design is employed to ensures a glitch free truly concurrent operation for both Host and DMX interfaces. By developing an application around a TRI command set, threading or task developments required to handle the timing of the RDM protocol will be greatly reduced or eliminated.

2.8 Firmware Upgrade

The TRI electronics and firmware have been carefully designed for ease of upgrade by means of non-intrusive programming (NIP). A method of NIP has been developed that sets this product apart from others preventing corrupt or incompatible firmware being loaded. The TRI products have been engineered with a robust system that will recover form, or complete an upgrade, even in the event of a power interruption during the process.

Lamb Park, Drewsteignton, Nr Exeter, Devon. UK EX6 6PN

3. Standard Command Format

3.1 Terminology

3.1.1 Host PC

The application employed or developed to utilise a TRI will be run on what will be referred to as the ***host*** or ***host PC***. A TRI will be connected to the host using a USB connection.

3.1.2 Request Frames

Frames generated by the application on the host PC and sent to the device via the USB connection will be referred to as ***request*** or ***requests***.

3.1.3 Response Frames

Frames returned from the device to the Host PC in response to a request frame will be referred to as a ***responses*** or ***response***.

3.1.4 Frame Exchange

For each Request Frame, a relevant response frame will be expected, without which an error condition will be present. The cycle of a request frame and its associated response frame will be referred to as an ***exchange.***

3.1.5 Command Identifier (CI)

The action or purpose of each frame exchange will be determined by a command parameter, which will be referred to as a ***CI***.

3.1.6 Return Code (RC)

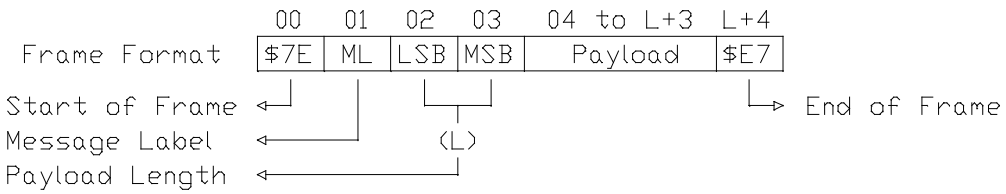
The first parameter in the Payload of each response frame will be a ***return code*** or ***RC***. The outcome of the exchange determines the value of the RC.

3.2 Overview

The standard frame format has been developed to be backward compatible with the Legacy frame Command format. Each request frame sent by the application to the device is expected to solicit a response frame. The Request and response frames are identical in format.

3.3 Frame Formats

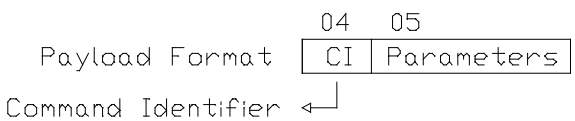
Below is a schematic of the frame format common to all request and response frames that comprise an exchange. For all Standard Format frames, the Message Label will be $58 ('X'). Legacy commands use the same Frame format with a variety of message labels.



The extension Message Label $58 ('X') identifies the frame to the TRI as a standard frame format.

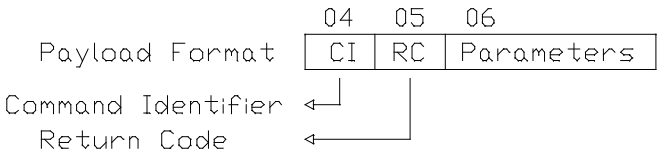
3.3.1 System Request Frames

All parameters following a command identifier in a request frame are elaborated according to the CI used. See CI elaborations for specific details.



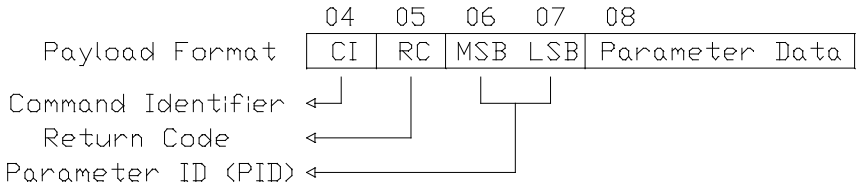
3.3.2 System Response Frames

Response Frame payloads start with the CI used in the request to solicit the response. The CI in a response frame is followed by a RC byte then any expected parameters. When the RC is non zero there will be no parameters.



3.3.3 RDM Get and Set Command Frames

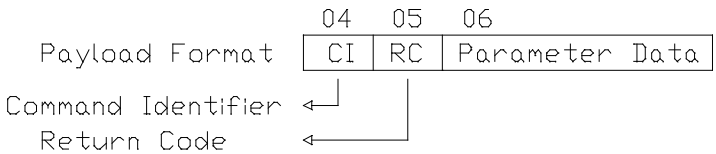
The Payload used for making RDM calls through a TRI is expanded from the System Request Frame in the diagram below for clarity. Refer to the CI_RemoteGet and CI_RemoteSet elaborations for more details.



Refer to the Table A-3 in Appendix A of the E1.20 – 2006 Standard for a list of PID values.
For queued messages refer to the RDM Message Frames below

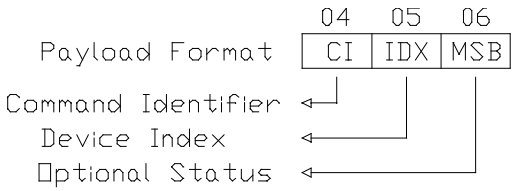
3.3.4 RDM Get and Set Response Frames

The payload of an RDM response frame is effectively the same as a System response frame.



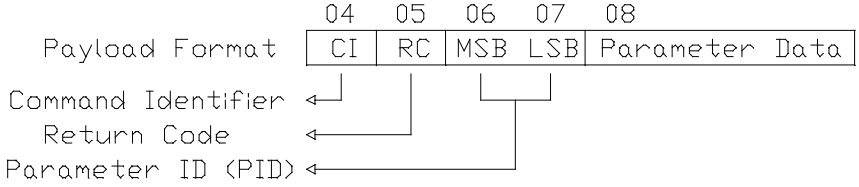
3.3.5 RDM Queued Message Command Frames

Queued messages do not invite the same PID response as the command and must therefore use the CI_QueuedGet command to retrieve them.



3.3.6 RDM Queued Message Response Frames

The PID of the response to the request preceeds the Parameters data (PD) in the response, otherwise the response frame is formatted the same as all other RDM response frames.



4. Legacy Command Format

Legacy commands differ from Standard command in several ways including:

- Some commands do not have a response
- No facility is available for error messages to be returned to the host application.

The Frame Format has the same structure as detailed in the Standard Frame Format with the message label determining the function

4.1 Summary

Function	ML	Description
Device Version, Identity and Settings		
ML_GetSerial	$0A	Returns the serial number of the device
ML_GetConfig	$03	Returns a packed list of labels, settings and data
ML_SetConfig	$04	Updates the device settings and data
ML_MakersRef	$4D	Returns the manufacturer ESTA assigned ID and name
ML_ModelName	$4E	Returns the model Identifier and name
DMX Input and Output Functions		
ML_SingleTX	$07	Transmits one frame without repetition
ML_RepeatTX	$06	Transmits or updates frame sent iteratively

4.2 Message Elaborations

4.2.1 ML_GetSerial

Purpose
To return the Serial Number of the TRI

Request Payload
Byte 0: $0A

Response Payload
Byte 0: $0A
Byte 1 to 4: BCD representation of serial, LSB to MSB

4.2.2 ML_GetConfig

Purpose
To return the device version details and current configuration settings and data

Request Payload
Byte 0: $03

Response Payload
Byte 0: $03
Byte 1 to 2: Firmware version, LSB to MSB
Byte 3: Break setting (BRK) starting each DMX Frame
Setting units are 10.67µs increments.
Valid unit values are 8 to 103

Byte 4: "Mark After Break" setting (MAB) in the transmission of DMX Frames
Setting units are 10.67µs increments.
Valid unit values are 1 to 103

Byte 5: DMX frame transmission rate.
Setting units are Frames/Second.
Valid unit values are 1 to 40

Byte 6 to Last: User Data
Host application specific data stored on the TRI

Notes
The TRI stores time durations as µs integer values. Values set using the standard command set will be rounded to the nearest integer value representing the 10.67µs increment.
The User data will only be returned when previously set using the ML_SetConfig
Re configuring the TRI will erase the User Data.

4.2.3 `ML_SetConfig`

Purpose
To update the device configuration settings and data

Request Payload
Byte 0: $04
Byte 1: Break setting (BRK) starting each DMX Frame
Setting units are 10.67µs increments.
Valid unit values are 9 to 103

Byte 2: Mark After Break setting (MAB) in the transmission of DMX Frames
Setting units are 10.67µs increments.
Valid unit values are 1 to 103

Byte 3: DMX frame transmission rate.
Setting units are Frames/Second.
Valid unit values are 1 to 40

Byte 4 to Last: User Data
Host application specific data stored on the TRI

Response Payload
N/A

Notes
The TRI stores time durations as µs integer values. The maximum duration settings have been restricted to 103 (11ms) to avoid paradox at maximum throughput.
The TRI does not respond to this call
Re configuring the TRI will erase the User Data.

4.2.4 `ML_MakersRef`

Purpose
To identify the manufacturer of the product in order to determine command set support

Request Payload
Byte 0: $4D

Response Payload
Byte 0: $4D
Byte 1 to 2: ESTA manufacturer assigned Identification, LSB to MSB
Byte 3 to Last: Manufacturer name ASCII string up to 32 bytes.

Notes
The JESE manufacturer assigned ID is $6864
The manufacturer name string is not null terminated.

4.2.5 `ML_ModelName`

Purpose
To identify the model from a manufacturer

Request Payload
Byte 0: $4E

Response Payload
Byte 0: $4E
Byte 1 to 2: Model Identification number, LSB to MSB
Byte 3 to Last: Model name ASCII string up to 32 bytes.

Notes
The model number for the DMX-TRI = $0001
The model number for the RDM-TRI = $0002
The Model name string is not null terminated.

4.2.6 `ML_SingleTX`

Purpose
To transmit a single DMX frame

Request Payload
Byte 0: $07
Byte 1: DMX Start code.
Byte 2 to Last: DMX Slot data / RDM Frame

Response Payload
N/A

Notes
The TRI does not respond to this call
If the device is in iterative transmission mode, the transmission will stop after the transmission of the frame submitted by this call.

4.2.7 `ML_RepeatTX`

Purpose
To iteratively transmit DMX frames at the configured rate.

Request Payload
Byte 0: $06
Byte 1: DMX Start code.
Byte 2 to Last: DMX Slot data

Response Payload
N/A

Notes
The TRI does not respond to this call
The device will leave the iterative frame transmission mode when it receives a command other than ML_RepeatTX.

4.3 Error Handling

There is no provision in the structure of the Legacy commands to report exceptions to the Host Application.
The TRI will respond to detected errors with a beep.

Lamb Park, Drewsteignton, Nr Exeter, Devon. UK EX6 6PN

5. Command Identifiers

5.1 Summary

The table below is a quick reference guide to the available CI to exchange with the a TRI. Refer to the elaboration for each CI for details on request and response parameter formatting.

Function	CI	Description
		Device Version and Identity
CI_NamePlate	$00	Returns the literal name of the product
CI_HW_VerGet	$01	Returns the revision level of the device hardware
CI_FW_VerGet	$02	Returns the version of firmware loaded in the device
CI_SerialGet	$03	Returns the serial number of device set at manufacture
CI_UniqueGet	$04	The current DMX UID being reported, default or overridden
CI_OverideID	$0C	Overrides the DMX UID reported by the device
CI_RestoreID	$0D	Restores the DMX UID reported by the device to default
		DMX Frame Settings
CI_GetFrRate	$11	Defines the rate at which DMX frames are transmitted in Frames / Second.
CI_SetFrRate	$12	
CI_GetMkTime	$13	Defines the Mark after Break (MAB) interval time in µs for the DMX frame transmission.
CI_SetMkTime	$14	
CI_GetBkTime	$15	Defines the Break (BRK) interval timing in µs for the DMX frame transmission.
CI_SetBkTime	$16	
CI_GetWaitRX	$17	Defines time in ms that the unit waits after an RDM command is issued before timing out and returning a response error.
CI_SetWaitRX	$18	
		DMX Input and Output Functions
CI_SingleTX	$21	Transmits or inserts one frame without repetition
CI_RepeatTX	$22	Transmits or updates frame sent repetitively
CI_FinishTX	$23	Stops any repetitive transmission on completion of frame
CI_ResumeTX	$24	Resumes previous iterative transmission
CI_SetLevel	$25	Changes a single slot value in repetitive frame transmission
CI_SingleRX	$28	Request the next DMX frame received by the device
		RDM – Remote Device Management
CI_ResetList †	$30	Clears device of all registered RDM line devices
CI_AddToList †	$31	Non automated addition of RDM line device to register
CI_DiscoMask †	$32	Single discovery frame sent with supplied mask
CI_DiscoAuto †	$33	Automated RDM discovery procedure
CI_DiscoStat †	$34	Status of Automated Discovery procedure
CI_RemoteUID †	$35	Returns the UID of an RDM line device in local device register
CI_RemoteGet †	$38	Query to remote line device listed in local device register
CI_RemoteSet †	$39	Instruction to remote line device listed in local device register
CI_QueuedGet †	$3A	Queued message collection from remote line device listed in register
CI_SetFilter †	$3D	Sets manufacturer's ID code to filter RDM broadcast commands
		Device Configuration and Upgrade
CI_SetAccess ✶	$40	Used to enter an access licence key
CI_LoadPage	$41	Uploads a 64 byte page of binary program data to device
CI_Reconfig	$42	Causes the device to reprogram from loaded binary data
CI_GetRegKey	$43	Used to save and retrieve configuration and licence data, binding the device to a host application
CI_SetRegKey	$44	
CI_FW_Build	$4F	The build number of the loaded binary program data

Note † Available to RDM-TRI and licensed DMX-TRI units. ✶ Available to unlicensed DMX-TRI units.

5.2 Command Elaborations

This section specifies the payload usage for each of the Command Identifiers. Refer to the Frame format section for the makeup of a frame with a payload.

5.2.1 `CI_NamePlate`

Purpose
Returns the literal name or unique identifier of the product

Request Parameters
Byte 0: Response data
Valid values are 0 to 1

Response Parameters
Dependant on Request Parameter Byte 0
When 0 =>
Byte 0 to Last: Null terminated ASCII string.

When 1 =>
2 byte, 16 bit product identifier, LSB to MSB

Notes
The returned data will be the same for all revisions of the product and will be unique for each type of product.

5.2.2 `CI_HW_VerGet`

Purpose
To return the version of the TRI Hardware

Request Parameters
None

Response Parameters
Null terminated ASCII String.
The string will be in the format VV.RR where VV is the major version and RR is the revision for the version

Notes
The hardware version is part of the Boot block code and will not change for the revision of the hardware.

5.2.3 `CI_FW_VerGet`

Purpose
To return the version of the Firmware loaded on the TRI

Request Parameters
Optional – Default = 0
Byte 0: Response format
Valid values are 0 to 1

Response Parameters
Dependant on Request Parameter Byte 0
When 0 =>
Byte 0: BCD Version Minor revision N°
Byte 1: BCD Version Major revision N°

When 1 =>
Null terminated ASCII String
The string will be in the format MM.SS.RRRR where MM is the major version, SS is the sub version and RRRR is the revision for the version.

5.2.4 `CI_SerialGet`

Purpose
To return the Serial Number of the TRI

Request Parameters
Optional – Default = 0
Byte 0: Response format
Valid values are 0 to 2

Response Parameters
Dependant on Request Parameter Byte 0
When 0 =>
4 byte, 32 bit number, LSB to MSB

When 1 =>
Null terminated ASCII String, maximum length of ten digits.

When 2 =>
5 byte BCD representation, LSB to MSB

Notes
The Serial number returned by this CI will not be affected when the UID is over-ridden.

5.2.5 `CI_UniqueGet`

Purpose
To return six bytes, representing the ESTA unique identity (UID) of the attached TRI.

Request Parameters
None

Response Parameters
Byte 0 to 1
Two bytes representing the Makers Identifier as allocated by ESTA

Byte 2 to 5
32 bit representation of the device's unique identification

Notes
Unless overridden, the above will consist of two bytes containing the Makers ID as allocated by ESTA ($6864) followed by 32bit representation of the devices serial N°, MSB to LSB. The UID returned in this command will be the same as that used by the device in interaction with other RDM devices.

5.2.6 `CI_UniqueSet`

Purpose
To change the identity used by the attached TRI when interacting with other RDM devices.

Request Parameters
Byte 0 to 5
6 bytes of the new UID to be used

Response Parameters
None

Notes
The overridden UID is volatile and must be re-set each time that the device is powered up.
When the device is powered up, the device will default to its native UID.

When this command is used, records of discovered devices will be erased to avoid addressing conflicts.

5.2.7 CI_RestoreID

Purpose
To revert to the native identity used by the attached peripheral device used when interacting with other RDM devices.

Request Parameters
None

Response Parameters
None

Notes
If this command causes the UID of the attached TRI to change, records of discovered devices will be erased to avoid addressing conflicts.

5.2.8 CI_GetFrRate

Purpose
To return the current setting for the iterative DMX frame transmission rate.

Request Parameters
None

Response Parameters
Byte 0: The Frame transmission rate in Frames/Second.
Byte 1: The minimum value supported by the device.
Byte 2: The maximum value supported by the device.

Notes
The minimum and maximum values are included to aid exception handling by the Host application.

5.2.9 CI_SetFrRate

Purpose
To update the current setting for the iterative DMX frame transmission rate.

Request Parameters
Byte 0: The new transmission rate in Frames/Second.

Response Parameters
None

5.2.10 CI_GetMkTime

Purpose
To return the current setting for the “Mark After Break” time (MAB) in the transmission of DMX Frames.

Request Parameters
None

Response Parameters
Values are 16 bit, LSB to MSB in µs
Byte 0 to 1: The current setting for the device
Byte 2 to 3: The minimum value supported by the device
Byte 4 to 5: The maximum value supported by the device

5.2.11 `CI_SetMkTime`

Purpose
To update the current setting for the "Mark After Break" time (MAB) in the transmission of DMX Frames

Request Parameters
Values are 16 bit, LSB to MSB in µs
Byte 0 to 1: The new Mark after Break period

Response Parameters
None

Notes
Request parameter must be in the range of 8 to 10,000.
If the combination of selected time parameters exceeds 100% of the duty cycle, an error code will be returned and the command denied.

5.2.12 `CI_GetBkTime`

Purpose
To return the current setting for the Break time (BRK) starting each DMX Frame.

Request Parameters
None

Response Parameters
All values are 16 bit, LSB to MSB in µs
Byte 0 to 1: The current setting for the device
Byte 2 to 3: The minimum value supported by the device
Byte 4 to 5: The maximum value supported by the device

5.2.13 `CI_SetBkTime`

Purpose
To update the current setting for the Break time (BRK) starting each DMX Frame.

Request Parameters
Values are 16 bit, LSB to MSB in µs
Byte 0 to 1: The new Break period

Response Parameters
None

Notes
Request parameter must be in the range of 88 to 1408. If the combination of selected time parameters exceeds 100% of the duty cycle, an error code will be returned and the command denied.

5.2.14 `CI_GetWaitRX`

Purpose
To return the current setting of the duration the TRI waits after it issues an RDM command to a responder before timing out.

Request Parameters
None

Response Parameters
Values are in ms
Byte 0: The current duration setting
Byte 1: The minimum supported value
Byte 2: The maximum supported value

Notes
See the CI_SetWaitRX for more details.

5.2.15 `CI_SetWaitRX`

Purpose
To set the duration that the TRI waits for a response from a responder after it issues a RDM command before it times out.

Request Parameters
Byte 0: The new duration in ms

Response Parameters
None

Notes
Under normal conditions, this value will be set at 0 for which the TRI employs the turn around timing criteria specified in the standard.
This CI is provided for testing and in some cases, to make the TRI interoperable with RDM equipment that does not respond to a command within the time specified by the E1.20 – 2006 standard. Where possible, contact the supplier of the remote equipment to resolve the response timing issues. When all such issues are resolved, this setting should be returned to 0.

5.2.16 `CI_SingleTX`

Purpose
To transmit a single start code and DMX frame when the device is idle or in iterative transmission mode.

Request Parameters
Byte 0: Options
The Option determines how the request is handled when sequential requests are made at a rate in excess of the DMX frame rate.
When 0 =>
The device will return from the request when it has been processed.

When 1 =>
The device will discard the request and return a busy RC.

Byte 1: DMX frame Start Code
Byte 2 to Last : DMX slot values

Response Parameters
None

Notes
When the device is in iterative transmission mode, the single frame will be inserted and iterative transmission resumed. In all other cases, the device will return to receive mode and be ready to buffer incoming frames.

5.2.17 `CI_RepeatTX`

Purpose
To initiate the iterative transmission of DMX frames or update the DMX iterative transmission.

Request Parameters
Byte 0: Option
The Option determines how the request is handled when sequential requests are made at a rate in excess of the DMX frame rate.
When 0 =>
The device will return from the request when it has been processed.

When 1 =>
The device will discard the request and return a busy RC.

Byte 1: DMX frame Start Code
Byte 2 to Last: DMX slot values

Response Parameters
None

Notes
Once requested, the device will iterate transmission of the DMX frame until instructed otherwise or loses power.

5.2.18 `CI_FinishTX`

Purpose
To suspend iterative frame transmission and set the device to receive mode.

Request Parameters
None

Response Parameters
None

Notes
If the TRI is in receive mode when this command is issued, it will have no effect.
When in iterative frame transmission mode, the frame currently being transmitted will complete before the unit returns to receive mode.

5.2.19 `CI_ResumeTX`

Purpose
To resume a suspended iterative frame transmission mode

Request Parameters
None

Response Parameters
None

Notes
If the device has not been suspended from iterative frame transmission mode, the TRI will be unable to comply and return a mode error RC.

5.2.20 `CI_SetLevel`

Purpose
To change a single slot value in the iterative frame transmission

Request Parameters
Byte 0 to 1: Slot number of the value to update, LSB to MSB (Starting at 1)
Byte 2: The new value for the slot in the DMX frame

Response Parameters
None

Notes
This operation will fail if the device has not entered iterative frame transmission mode and return a mode error RC. If the slot number is not in the range of the universe last transmitted, the command will fail and a constraint error RC will be returned.

5.2.21 `CI_SingleRX`

Purpose
To provide the means for incoming DMX frames to be retrieved from the device as they are received, or from the buffer. DMX frames with any start code may be received with this command.

Request Parameters
Byte 0: Response option
Valid values for the option are 0 to 3

When 0 =>
Unconditional fetch, no time out, no update detection.
The call will return without waiting with then next frame in sequence if available.

When 1 =>
Conditional fetch, no time out, detects updates.
The call will return without waiting, with the next updated frame if available.

When 2 =>
Conditional fetch, will time out, no update detection.
The call will wait up to a second for the next frame in sequence and return with the frame is available.

When 3 =>
Conditional fetch, will time out, detects updates.
The call will wait up to 1 a second for the next updated frame. If no updated frame is available on timeout, the next frame in sequence will be returned if available.

Response Parameters

Byte 0: Status Code
The expected values for the Status Code are as follows:
When 0 =>
Normal response, no overrun.

When 1 =>
Normal response, frame buffer has overrun.

When 2 =>
No complete frame ready in buffer.

When 3 =>
No DMX received in the last second

Byte 1: DMX frame Start Code
Byte 2 to Last: DMX Slot values

Notes

As the number of slots in a DMX frame may only be determined at the start of the following frame, there will be a degree of latency in returning the frame, depending on rate at which frames are transmitted.

By using response options without time out detection, the host application may employ the command as part of a pseudo real-time task, to monitor the receive buffer. Response Status code 3 will indicate when the device is not receiving DMX frames at the minimum rate specified by the standard.

5.2.22 `CI_ResetList`

Purpose

To clear the list in the TRI of registered RDM devices added in the process of a discovery or manually

Request Parameters

None

Response Parameters

None

Notes

This CI is available to RDM-TRI and licensed DMX-TRI units only.
The use of this command will un-mute all compliant RDM devices, ready for a Host driven discovery if required.

5.2.23 `CI_AddToList`

Purpose
To cause the attached device to register a single RDM compliant device for use with related commands.

Request Parameters
Byte 0 to 5: The 6 bytes of the target device UID

Response Parameters
Byte 0: Device Index
The ordinal index number for the assigned device. If zero is returned, the target device was not detected and not added to the list.

Notes
This CI is available to RDM-TRI and licensed DMX-TRI units only.
This command may be used to quickly register a responder device with without going through the discovery process. The “Get Progress” Command does not work in conjunction with this command.
The successfully registered device will be muted.

5.2.24 `CI_DiscoMask`

Purpose
To provide a means of RDM discovery without using the automated process provided by the TRI.
The command causes a Discovery Unique Branch to be run

Request Parameters
Bytes 0 to 5: Lower bound UID Bit Mask
Bytes 6 to 11: Upper bound UID Bit Mask

Response Parameters
Byte 0: Status Code
When 0 =>
No response was received in reply to the discovery

When 1 =>
One response was received in reply to the discovery

When 2 =>
More than one response was received in reply to the discovery

If the Status Code = 1 then
Bytes 1 to 6: UID of the responding device.

Notes
This CI is available to RDM-TRI and licensed DMX-TRI units only.
If the remote device is already registered, it will be muted and will not respond to this command.
To mute a discovered device, use the AddToList CI.

5.2.25 `CI_DiscoAuto`

Purpose
To autonomously discover and register all RDM compliant devices.

Request Parameters
None

Response Parameters
None

Notes
This CI is available to RDM-TRI and licensed DMX-TRI units only.
The TRI will return from this call immediately whilst the on board process expedites the discovery routine.

The time taken to complete the process will depend on the number of devices attached and the number of devices from different manufacturers of the attached devices.
Progress of the discovery process can be tracked in real time using the DiscoStat CI.

5.2.26 `CI_DiscoStat`

Purpose
To track the progress of the automated device discovery process initiated by the "DiscoAuto" command

Request Parameters
None

Response Parameters
Byte 0: The number of unique devices discovered so far
Byte 1: The current status of the process
When 0 =>
The process is inactive or has completed.

When 1 =>
Discovery process is in progress.

Notes
This CI is available to RDM-TRI and licensed DMX-TRI units only.
This more frequently this command is used during a discovery process, the more overhead will be place on the device process, adding to the time taken to complete the discovery.
This command may be used at any time to ascertain the number of devices registered with the TRI.

5.2.27 `CI_RemoteUID`

Purpose
To return the Unique Identifier of a remote RDM compliant device from the register of the TRI

Request Parameters
Byte 0: Index Number of the remote device.
Must be in the range of discovered devices in the register.

Response Parameters
Byte 0 to 5: The 6 bytes of the target device UID

Notes
This CI is available to RDM-TRI and licensed DMX-TRI units only.
Each responder device registered with the TRI is referenced by its index number. This CI accesses the register in the TRI.

5.2.28 `CI_RemoteGet`

Purpose
To provide a universal and simplified interface to all RDM non-modifier commands where the response PID is the same as the request PID.

Request Parameters
Byte 0: Index Number of the remote device.
Must be in the range of discovered devices in the register.

Byte 1 to 2:
Sub Device Index, MSB to LSB

Byte 3 to 4:
Parameter ID (PID), MSB to LSB

Optional Parameters
Bytes 5 to Last
Parameter Data (PD)

Response Parameters
Byte 0 to Last:
Parameter Data (PD)

Notes
This CI is available to RDM-TRI and licensed DMX-TRI units only.
For the retrieval of queued messages (QUEUED_MESSAGE), refer to CI_QueuedGet
The Index number for a remote device is the ordinal number of the device listed in the TRI register.

5.2.29 `CI_RemoteSet`

Purpose
To provide a universal and simplified interface to all RDM modifier command.

Request Parameters
Byte 0: Index Number of the remote device.
Must be in the range of discovered devices in the register or zero for broadcast

Byte 1 to 2:
Sub Device Index, MSB to LSB

Byte 3 to 4:
Parameter ID (PID), MSB to LSB

Byte 5 to Last
Set Parameter Data (PD)

Response Parameters
Byte 0 to Last:
Return Parameter Data (PD)

Notes
This CI is available to RDM-TRI and licensed DMX-TRI units only.
The Index number for a remote device is the ordinal number of the device listed in the TRI register.
Index zero will cause the TRI to broadcast message to all devices, as per the filter setting. See CI_SetFilter to mask broadcast messages to Manufacturer specific equpment.
The majority of PID used in CI_RemoteSet do not produce any response parameters.

5.2.30 `CI_QueuedGet`

Purpose
To provide retrieve messages from the message queue of a remote device where PID QUEUED_MESSAGE would be used.

Request Parameters
Byte 0: Index Number of the remote device.
Must be in the range of discovered devices in the register.

Optional Parameter
Byte 1
Status Type, as defined in table A-4 of the standard.
Default value is STATUS_ADVISORY

Response Parameters
Byte 0 to 1:
Parameter ID (PID), MSB to LSB

Byte 2 to Last:
Parameter Data (PD)

Notes
This CI is available to RDM-TRI and licensed DMX-TRI units only.
The Index number for a remote device is the ordinal number of the device listed in the TRI register.

5.2.31 `CI_SetFilter`

Purpose
To set the two-byte ESTA Manufacturer ID for which RDM broadcast messages will be sent.

Request Parameters
Bytes 0 to 1:
ESTA Manufacture ID or $FFFF for all stations.

Response Parameters
None

Notes
This CI is available to RDM-TRI and licensed DMX-TRI units only.
RDM SET commands are issued by indexing device zero using CI_RemoteSet.
The default value after power up is $FFFF

5.2.32 `CI_LoadPage`

Purpose
To upload pages of data configuration data to the TRI for upgrading the firmware.

Request Parameters
Bytes 0 to 67: Firmware configuration page.

Response Parameters
None

Notes
Each page is checked for integrity as it is loaded. Complete integrity checking is not done till the Reconfig command is called.
For details on formatting the *.fud (field upgrade for devices) into configuration pages contact support@jese.co.uk

5.2.33 `CI_Reconfig`

Purpose
To command the TRI to reconfigure itself with new firmware loaded using the LoadPage CI

Request Parameters
None

Response Parameters
None

Notes
This command will take approximately 20 seconds to complete, depending on the size of the data being configured.
A compatible and error free file must be loaded in its entirety before this command is used.
Any inconsistencies will cause the command to fail and report the relevant error RC
On completion of a successful reconfiguration, the TRI will re-boot.

5.2.34 `CI_GetRegKey`

Purpose
To return the current configuration or licence data associated with a controlling application

Request Parameters
Byte 0: Page Number

Response Parameters
Byte 0 to Last: Configuration Data

Notes
The returned page size will be the same length as the data set for that page and limited to 256 bytes

5.2.35 `CI_SetRegKey`

Purpose
To set or update the configuration or licence data that associated with a controlling application.

Request Parameters
Byte 0: Page Number
Byte 1 to Last: Configuration data

Response Parameters
None

Notes
Maximum page size is limited to 256 bytes

5.2.36 `CI_FW_Build`

Purpose
To retrieve the Firmware build number.

Request Parameters
None

Response Parameters
Bytes 0 to 3 : 32 bit Build Number LSB to MSB

Notes
The Firmware build number will be the same as the Software Version ID reported in the RDM DEVICE_INFO

Lamb Park, Drewsteignton, Nr Exeter, Devon. UK EX6 6PN

6. Return Codes (RC)

6.1 Code Summary

Condition	RC	Description
Codes generated by the TRI		
RC_NoError	$00	Nothing to report
RC_Constraint	$01	Parameter data value is out of expected range
RC_CommandInst	$02	The CI used is not recognised or implemented
RC_NotAnOption	$03	The provided option is not available
RC_FrameFormat	$04	Received frame incorrectly formatted
RC_DataTooLong	$05	Frame length longer than expected
RC_DataMissing	$06	Frame missing expected data
RC_SystemMode	$07	Not allowed in current mode
RC_SystemBusy	$08	Not able to accept command at this time
RC_DataCheckSum	$0A	Data Check Sum failed
RC_Incompatible	$0B	Data Compatibility Error
Codes generated by the RDM Protocol		
RC_ResponseTime	$10	Request serviced by Message Queue (ACK_TIMER)
RC_ResponseWait	$11	One or messages are waiting in Message Queue
RC_ResponseMore	$12	Additional Data available after this response
RC_ResponseTran	$13	Incorrect Transaction number in response
RC_ResponseSubd	$14	Response from wrong sub device
RC_ResponseFmat	$15	Format of response not recognised
RC_ResponseCSum	$16	Response Checksum Error
RC_ResponseNone	$18	Expected response missing or timed out
RC_ResponseIdnt	$1A	Wrong device responding to request
RC_ResponseMute	$1B	Device Discovery Mute Error
RC_RespomseDisc	$1C	Duplicated or erroneous device detected
RC_ResponseUnEx	$1D	A response was received when not expected
Codes generated by a remote device		
NR_UnknownPID	$20	Remote device does not recognise the PID
NR_FormatError	$21	Remote device does not recognise frame format
NR_HardwareFault	$22	Remote device not able to comply due to a hardware fault
NR_ProxyReject	$23	Remote device not able to comply as a proxy
NR_WriteProtect	$24	Remote device is not able to accept a set command at this time
NR_UnsupportedCC	$25	Remote device does not allow data to be set with this PID
NR_OutOfRange	$26	Remote device value is out of expected range
NR_BufferFull	$27	Remote device not able to buffer data
NR_FrameOverflow	$28	Remote device not able to handle frame length
NR_SubdevUnknown	$29	Remote device – sub device not present