ENTTEC Pty/Ltd
ACN 089 252 791
ABN 94 388 357 479
Australia

Tel +61 3 9763 5755
Fax +61 3 9763 5688
Web: www.enttec.com

USB PRO MK2 API Specification

Version	1.3
Last Updated:	24/08/15

Purpose

This document specifies the interface requirements for computer based application programs to use the USB PRO Mk2 to send or receive DMX512 or RDM packets.

In this document, DMX USB Pro1 refers to the first generation DMX USB Pro which has one DMX port, and USB PRO Mk2 refers to the second generation DMX USB Pro which has two DMX ports.

Computer Setup

Install the FTDI USB device driver on the PC or Mac from here: www.ftdichip.com

To obtain the highest data transfer rates, the D2XX driver should be used in preference to the VCP driver. When using the VCP, the baudrate setting used to open the COM port is a dummy value, and does not control the USB communication speed.

Application Message Format

The application program communicates with the Widget via the FTDI driver using the format listed in the table below:

Size In Bytes	**Description**
1	Start of message delimiter, hex 7E.
1	Label to identify type of message.
1	Data length LSB. Valid range for data length is 0 to 600.
1	Data length MSB.
data_length	Data bytes.
1	End of message delimiter, hex E7.

Note that all integer fields in API messages must be stored in little endian order, with the least significant byte at the lowest address of the field in the message.

ENTTEC Pty/Ltd
ACN 089 252 791
ABN 94 388 357 479
Australia

Tel +61 3 9763 5755
Fax +61 3 9763 5688
Web: www.enttec.com

USB PRO Mk2 Status LED

The colour of the LED indicates the widget status, as follows:

Colour	Meaning
Flashing White	Firmware is running normally.
Steady Red	Firmware has bad CRC. To fix this, update the firmware.
Flashing Red	Bad or unrecognisable API message was received.
Steady Green	DMX/RDM packet sent on port 1.
Flashing Green	DMX/RDM packet received on port 1.
Steady Yellow	DMX/RDM packet sent on port 2.
Flashing Yellow	DMX/RDM packet received on port 2.
Steady Purple	MIDI data sent on MIDI output port.
Flashing Purple	MIDI data received on MIDI input port.
Steady Blue.	Standalone show is playing.

Backward Compatibility

All application programs which were written to work with the DMX USB Pro1 should work without change with the USB PRO Mk2, with the following exceptions:

- The USB PRO Mk2 has a single firmware version to support DMX and RDM, whereas the DMX USB Pro1 had separate firmware versions for DMX and RDM support. This means that querying the firmware version of the USB PRO Mk2 no longer indicates whether RDM is supported or not.
- The second DMX port must be left disabled, to avoid the USB PRO Mk2 generating messages other than those which a DMX USB Pro1 would generate.

By default, apart from two new messages (Set API Key and Query Hardware Version), the USB PRO Mk2 supports the same application messages as the USB Pro1. To enable support for all messages in this API document, an API key must be used to unlock the extended features of the USB PRO Mk2.

To detect if a DMX USB Pro1 or a USB PRO Mk2 is present, the Query Hardware Version message should be used. The DMX USB Pro1 will not reply to the Query Hardware Version request.

Please keep in mind, the Hardware Version value may change in the future to indicate a new revision of USB PRO Mk2.

Application Messages

1. Get Port Widget Parameters Request (Label=3 for port 1, Label=189 for port 2)

This message requests the non-volatile Widget configuration for one DMX port.

Size In Bytes	Description
1	LSB of user configuration size in bytes. Valid range for user configuration size is 0 to 508.
1	MSB of user configuration size in bytes.

2. Get Port Widget Parameters Reply (Label=3 for port 1, Label=189 for port 2)

The Widget sends this message to the PC in response to the Get Port Widget Parameters request.

Size In Bytes	Description
1	Firmware version LSB. Valid range is 0 to 255.
1	Firmware version MSB. Valid range is 0 to 255.
1	DMX output break time in 10.67 microsecond units. Valid range is 9 to 127.
1	DMX output Mark After Break time in 10.67 microsecond units. Valid range is 1 to 127.
1	DMX output rate in packets per second. Valid range is 1 to 40, or 0 for fastest rate possible.
user_configuration_size	User defined configuration data. See Set Port Widget Parameters request.

3. Set Port Widget Parameters Request (Label=4 for port 1, Label=195 for port 2)

This message sets the non-volatile Widget configuration for one DMX port, which is preserved when the Widget loses power.

Size In Bytes	Description
1	LSB of user configuration size in bytes. Valid range for user configuration size is 0 to 508.
1	MSB of user configuration size in bytes.
1	DMX output break time in 10.67 microsecond units. Valid range is 9 to 127.
1	DMX output Mark After Break time in 10.67 microsecond units. Valid range is 1 to 127.
1	DMX output rate in packets per second. Valid range is 1 to 40, or 0 for fastest rate possible (this will make the most difference when the output universe size is smallest).
user_configuration_size	User defined configuration data.

4. Received DMX Packet (Label=5 for port 1, Label=141 for port 2)

The Widget sends this message to the PC unsolicited, whenever the Widget receives a DMX or RDM packet from a DMX port, and the Receive DMX on Change mode for that DMX port is 'Send always'.

Size In Bytes	Description
1	DMX receive status. When this is 0, the DMX data in this message is valid. When this is nonzero, the DMX data in this message is

Size In Bytes	Description
	corrupted. Bit 0: 0=No error,1=Widget receive queue overflowed. Bit 1: 0=No error,1=Widget receive overrun occurred.
1 to 513	Received DMX data beginning with the start code. The size of the received DMX data can be determined from the overall message size.

5. Output Only Send DMX Packet Request (Label=6 for port 1, Label=203 for port 2)

This message requests the Widget to periodically send a DMX packet out of one of the Widget DMX ports at the configured DMX output rate for that DMX port. This message causes the Widget to leave the DMX port direction as output after each DMX packet is sent, so no DMX packets will be received as a result of this request.

If a show is playing on the input port, this request will stop the show playing.

The request will be ignored if the port is disabled (see Set Port Assignment request).

Size In Bytes	Description
25 to 513	DMX data to send, beginning with the start code. The overall message size specifies the size of the DMX data to send, and also sets the universe size (the number of DMX channels which are output).

The periodic DMX packet output will continue until the Widget receives any request message that changes the DMX port direction to input. The request messages that change the DMX port direction to input are:

Receive DMX ON Change Request,
Reprogram Firmware Request,
Send RDM Discovery Request if no show is playing on the port in the request,
Send RDM Packet Request if no show is playing on the port in the request.

6. Send RDM Packet Request

(Label=7 for port 1, Label=139 for port 2)

This message requests the Widget to send an RDM packet out of one of the Widget DMX ports, and then change the DMX port direction to input, so that RDM or DMX packets can be received.

If a show is playing on the input port, the show will resume playing after this request.

The request will be ignored if the port is disabled (see Set Port Assignment request).

Size In Bytes	Description
1 to 513	RDM data to send, beginning with the start code. The overall message size specifies the size of the RDM data to send.

7. Receive DMX ON Change (Label = 8 for port 1, Label=143 for port 2)

This message requests the Widget to send a DMX packet to the PC only when the DMX values change on the input DMX port.

By default the widget will always send the whole DMX packet, if you want to send on change it must be enabled by sending this message.

This message also reinitializes the DMX receive processing, so that if change of state reception is selected, the initial received DMX data is cleared to all zeros.

Size In Bytes	Description
1	0: Receive every DMX/RDM packet 1: Receive DMX changes only. RDM packets can NOT be received in this mode.

8. Received DMX Change Of State Packet (Label=9 for port 1, Label=132 for port 2)

The Widget sends one or more instances of this message to the PC unsolicited, whenever the Widget receives a changed DMX packet from a DMX port, and the Receive DMX on Change mode for that DMX port is 'Send on data change only'.

Size In Bytes	Description
1	Start changed byte number.
5	Changed bit array, where array bit 0 is bit 0 of first byte and array bit 39 is bit 7 of last byte.
1 to 40	Changed DMX data byte array. One byte is present for each set bit in the Changed bit array.

The user program can decode the message into a 513 byte received DMX data array, beginning with the start code. The algorithm to do this is shown below:

```
On startup, zero out the 513 byte received_dmx_array
For each Change Of State packet received
   changed_byte_index = 0
   For bit_array_index = 0 to 39
      If changed_bit_array[bit_array_index] is 1 then
         received_dmx_array[start_changed_byte_number * 8 + bit_array_index] =
            changed_dmx_data_array[changed_byte_index]
         Increment changed_byte_index
      Endif
   Endfor
Endfor
```

9. Get Widget Serial Number Request (Label = 10, no data)

This message requests the Widget serial number, which should be the same as that printed on the Widget case.

10. Get Widget Serial Number Reply (Label = 10)

The Widget sends this message to the PC in response to the Get Widget Serial Number request.

Size In Bytes	Description
4	BCD serial number, with LSB stored at lowest address.

11. Send RDM Discovery Request (Label=11 for port 1, Label=167 for port 2)

This message requests the Widget to send an RDM Discovery Request packet out of one of the Widget DMX ports, and then receive an RDM Discovery Response (see Received DMX Packet).

If a show is playing on the input port, the show will resume playing after this request.

The request will be ignored if the port is disabled (see Set Port Assignment request).

Size In Bytes	Description
38	DISC_UNIQUE_BRANCH RDM request packet to send.

12. RDM Controller Receive Timeout (Label=12 for port 1, Label=138 for port 2, no data)

The RDM controller received no reply message within receive timeout time (This message was added in DMX USB Pro1 firmware v2.4).

This message will only be generated when a Widget DMX port is being used as an RDM controller.

The timeout message will follow the **RDM discovery** reply message, whether or not the reply is partial or complete.

The timeout message will follow the RDM reply message (GET or SET), only when the reply is incomplete or unrecognizable.

13. Set API Key Request (Label = 13)

This message enables either the DMX USB Pro1 API (API1) messages or the USB PRO Mk2 API (API2) messages.

Labels in the range 1 to 12 inclusive, belong to API1. All labels in this document belong to API2.

When the USB PRO Mk2 is powered on, only the API1 messages are enabled. To enable API2 messages a magic number must (**BFC77FA4**) be sent.

Each time the USB PRO Mk2 receives any message which is not enabled, the message will be discarded and the Red LED will blink to show that the message was rejected.

Size In Bytes	Description
4	API key, with LSB stored at lowest address. The API key value to enable API1 is hex 0. The API key value to enable API2 is hex **BFC77FA4**.

LSB Storage means the last byte of the key is sent FIRST and the first bytes sent LAST

14. Query Hardware Version Request (Label = 14, no data)

This message requests the hardware version number of the USB PRO Mk2. The request is supported whether or not the USB PRO Mk2 firmware is valid.

The DMX USB Pro1 will not reply to this request, since the DMX USB Pro1 does not support the request.

15. Query Hardware Version Reply (Label = 14)

The Widget sends this message to the PC in response to the Query Hardware Version request.

Size In Bytes	Description
1	Hardware version. The following are valid values for USB PRO MK2 2: DMX USB PRO MK2 3: DMX USB PRO MK2 revB

16. Get Port Assignment Request (Label =144, no data)

This message requests the DMX/MIDI port assignment.

17. Get Port Assignment Reply (Label =144)

The Widget sends this message to the PC in response to the Get Port Assignment request.

Size In Bytes	Description
1	DMX port 1 assignment. 0: DMX port disabled.

Size In Bytes	Description
	1: DMX port enabled for DMX and RDM.
1	DMX port 2, MIDI IN port and MIDI OUT port assignment. 0: Ports disabled. 1: DMX port enabled for DMX and RDM. 2: MIDI IN and MIDI OUT ports enabled.

When using the MIDI IN port, the DMX port 2 should not be driven, otherwise the DMX signal could corrupt the MIDI IN data.

18. Set Port Assignment Request (Label =151)

This message sets the Widget DMX port assignment, which is not preserved when the Widget loses power.

Size In Bytes	Description
1	DMX port 1 assignment. 0: Port disabled. 1: Port enabled for DMX and RDM.
1	DMX port 2, MIDI IN port and MIDI OUT port assignment. 0: Ports disabled. 1: DMX port enabled for DMX and RDM. 2: MIDI IN and MIDI OUT ports enabled.

19. Received MIDI (Label =164)

The Widget sends this message to the PC unsolicited, whenever the Widget receives data on the MIDI IN port, and the MIDI IN port is enabled.

Size In Bytes	Description
1 to 512 (normally 1)	Received MIDI data bytes.

20. Send MIDI Request (Label =152)

This message requests the Widget to send MIDI data out of the MIDI OUT port.

Size In Bytes	Description
1 to 64	MIDI data to send. The overall message size specifies the size of the MIDI data to send.

21. Show Query Request (Label =198, no data)

This message requests the size of the non-volatile show storage memory.

22. Show Query Reply (Label =198)

The Widget sends this message to the PC in response to the Show Query request.

Size In Bytes	Description
4	Size of the non-volatile show storage memory in bytes.
1	Memory busy status. 0: Erase or write has completed. 1: Erase or write is in progress.

23. Show Block Erase Request (Label =148)

This message erases one block of the non-volatile show storage memory, so that all bytes are set to hex FF. The block size is 64 kbytes for all blocks, except for the first block which may be smaller. The show storage memory must be erased once before a new show can be written to the show storage memory.

After the PC issues a show erase request, the PC should issue show query requests, to detect when the erase has finished. This is necessary to prevent the PC issuing another show erase request or show write request while the previous erase or write request is still in progress, which could cause loss of API messages.

Size In Bytes	Description
4	Subcommand character array, set to 'ERAS'.
2	Block number to erase. Valid range is 0 to (number of show memory blocks – 1), where the number of show memory blocks is the integer (size from show query reply + hex FFFF) / hex 10000.

24. Show Sector Erase Request (Label =148)

This message erases one sector of the non-volatile show storage memory, so that all bytes are set to hex FF. The sector size is 4 kbytes. The show storage memory must be erased once before a new show can be written to the show storage memory.

After the PC issues a show erase request, the PC should issue show query requests, to detect when the erase has finished. This is necessary to prevent the PC issuing another show erase request or show write request while the previous erase or write request is still in progress, which could cause loss of API messages.

Size In Bytes	Description
4	Subcommand character array, set to 'ERSE'.
2	Sector number to erase. Valid range is 0 to (number of show memory sectors – 1), where the number of show memory sectors is the integer (size from show query reply) / hex 1000.

25. Show Write Request (Label =148)

This message writes data to the non-volatile show storage memory. The show storage memory must be erased once before a new show can be written to the show storage memory.

After the PC issues a show write request, the PC should issue show query requests, to detect when the write has finished. This is necessary to prevent the PC issuing another show erase request or show write request while the previous erase or write request is still in progress, which could cause loss of API messages.

See Show File Format section for a description of the show format.

Size In Bytes	Description
4	Subcommand character array, set to 'WRIT'.
4	Address. Valid range is 0 to (size from show query reply – number of data bytes to write).
1 to 256	Data to write.

26. Show Read Request (Label =130)

This message reads data from the non-volatile show storage memory.

Size In Bytes	Description
4	Address. Valid range is 0 to (size from show query reply – number of data bytes to read).
2	Number of bytes to read, with LSB stored at low address. Valid range is 1 to 256.

27. Show Read Reply (Label =130)

The Widget sends this message to the PC in response to the Show Read request. See Show File Format section for a description of the show format.

Size In Bytes	Description
4	Address from show read request.
1 to 256	Data read.

28. Start Show Request (Label =148)

This message starts replay of the stored show if the CRC of the stored show header record is good.

Size In Bytes	Description
4	Subcommand character array, set to 'STAR'.

29. Stop Show Request (Label =148)

This message stops replay of the stored show.

Size In Bytes	Description
4	Subcommand character array, set to 'STOP'.

RDM Implementation Details

Enttec has an RDM protocol PC library that works with the Widget, for creating RDM capable PC applications.

The RDM protocol is a half duplex request/response protocol. To comply with the RDM protocol, the RDM Responder must send only a single response message in reply to each request message from the RDM Controller.

The two RDM ports are considered to be separate RDM devices, as each RDM port is assigned a different UID. Each UID consists of the Enttec manufacturer ID concatenated with a number derived from the Widget Serial Number, as follows:

UID1[0]=hex 45, UID1[1]=hex 4E, UID1[2]=N1[3], UID1[3]=N1[2],
UID1[4]=N1[1], UID1[5]=N1[0].

UID2[0]=hex 45, UID2[1]=hex 4E, UID2[2]=N2[3], UID2[3]=N2[2],
UID2[4]=N2[1], UID2[5]=N2[0].

where
N1 = 2 * SERIAL_NUMBER,
N2 = N1 + 1.

There is a special message with its own label for sending the RDM Discovery request, to enable the Widget to receive the discovery response, which has no break unlike all other types of RDM packet. The Send RDM request should be used to send any RDM packet other than the RDM discovery request.

The behaviour of the RDM Responder Widget is specified in the following table.

Request Received By RDM Responder Widget	**Reply Sent By RDM Responder Widget**	**Request Message Passed To PC**
CC=DISCOVERY_COMMAND PID=DISC_UNIQUE_BRANCH	Discovery response when Widget UID is inside range of UIDs in request message and RDM discovery has not been muted.	No
CC=DISCOVERY_COMMAND PID=DISC_MUTE	DISC_MUTE response when request message was not broadcast.	No
CC=DISCOVERY_COMMAND	DISC_UN_MUTE response when request message was	No

Request Received By RDM Responder Widget	**Reply Sent By RDM Responder Widget**	**Request Message Passed To PC**
PID=DISC_UN_MUTE	not broadcast.	
CC=GET_COMMAND or CC=SET_COMMAND	ACK_TIMER response with time of 0.1 seconds.	Yes. PC should respond with Queued message.
CC=GET_COMMAND PID=QUEUED_MESSAGE	Queued message (see ACK_TIMER).	No
Any unrecognised message or message with bad RDM checksum.	None	Yes

The RDM Responder Widget has space to store a single queued RDM message only, per DMX port. The ACK_TIMER and queued message mechanism ensure that the timing requirements of the RDM specification can be met, even though the PC application is slow to respond due to USB latency or scheduling delays for example.

Show File Format

This section specifies the format of the single show file that can be stored and replayed by the USB PRO Mk2. The show file consists of the following consecutive sections:

Header record.
Frame records.

1. Header Record

Size In Bytes	Description
4	File format version, set to 'SFA1'.
128	Show name text to identify the file. This field is not used by the USB Pro2.
4	Number of frame records, with LSB stored at low address. Minimum valid value is 1.
2	Number of DMX slots per frame record. Valid range is 1 to 512.
1	Bit 0: DMX port to play show on, where 0=DMX1, 1=DMX2. Bit 1: Play show at power on, where 0=disabled, 1=enabled. Bit 2: Backup enable, where 0=disabled, 1=enabled. When backup is enabled, DMX1 will pass through to DMX2, and loss of DMX on DMX1 will start the show on DMX2. Show is invalid if Backup enable is set and DMX port is DMX1.
1	Delay before looping, in seconds. Valid range is 0 to 255.
2	Loop count, with LSB stored at low address. Valid range is 0 to 65534. The show will play (Loop count + 1) times. 65535 is loop forever.
112	Unused, set to hex FF.
2	CRC calculated over rest of header record, with LSB stored at low address.

The CRC is calculated using the XMODEM CRC polynomial (hex 1021), and initial CRC value of 0.

The CRC is used to detect corruption of the record. If the CRC is bad, the show will not play.

2. Frame Record

Each frame record specifies when to play a DMX frame in the show, as well as the DMX data to output for one scene of the show. The number of DMX slots output, excluding the DMX start code, is specified in the header record.

Size In Bytes	**Description**
4	Time in milliseconds, with LSB stored at low address.
1 to 512	DMX data bytes.
2	CRC calculated over rest of frame record, with LSB stored at low address.

The CRC is calculated using the XMODEM CRC polynomial (hex 1021), and initial CRC value of 0.

The CRC is used to detect corruption of the record. If the CRC is bad, the show will stop playing when the bad frame record is read.

ENTTEC Pty/Ltd
ACN 089 252 791
ABN 94 388 357 479
Australia

Tel +61 3 9763 5755
Fax +61 3 9763 5688
Web: www.enttec.com

USB PRO MK2 API Specification

Version	1.3
Last Updated:	24/08/15

Purpose

This document specifies the interface requirements for computer based application programs to use the USB PRO Mk2 to send or receive DMX512 or RDM packets.

In this document, DMX USB Pro1 refers to the first generation DMX USB Pro which has one DMX port, and USB PRO Mk2 refers to the second generation DMX USB Pro which has two DMX ports.

Computer Setup

Install the FTDI USB device driver on the PC or Mac from here: www.ftdichip.com

To obtain the highest data transfer rates, the D2XX driver should be used in preference to the VCP driver. When using the VCP, the baudrate setting used to open the COM port is a dummy value, and does not control the USB communication speed.

Application Message Format

The application program communicates with the Widget via the FTDI driver using the format listed in the table below:

Size In Bytes	**Description**
1	Start of message delimiter, hex 7E.
1	Label to identify type of message.
1	Data length LSB. Valid range for data length is 0 to 600.
1	Data length MSB.
data_length	Data bytes.
1	End of message delimiter, hex E7.

Note that all integer fields in API messages must be stored in little endian order, with the least significant byte at the lowest address of the field in the message.

ENTTEC Pty/Ltd
ACN 089 252 791
ABN 94 388 357 479
Australia

Tel +61 3 9763 5755
Fax +61 3 9763 5688
Web: www.enttec.com

USB PRO Mk2 Status LED

The colour of the LED indicates the widget status, as follows:

Colour	Meaning
Flashing White	Firmware is running normally.
Steady Red	Firmware has bad CRC. To fix this, update the firmware.
Flashing Red	Bad or unrecognisable API message was received.
Steady Green	DMX/RDM packet sent on port 1.
Flashing Green	DMX/RDM packet received on port 1.
Steady Yellow	DMX/RDM packet sent on port 2.
Flashing Yellow	DMX/RDM packet received on port 2.
Steady Purple	MIDI data sent on MIDI output port.
Flashing Purple	MIDI data received on MIDI input port.
Steady Blue.	Standalone show is playing.

Backward Compatibility

All application programs which were written to work with the DMX USB Pro1 should work without change with the USB PRO Mk2, with the following exceptions:

- The USB PRO Mk2 has a single firmware version to support DMX and RDM, whereas the DMX USB Pro1 had separate firmware versions for DMX and RDM support. This means that querying the firmware version of the USB PRO Mk2 no longer indicates whether RDM is supported or not.
- The second DMX port must be left disabled, to avoid the USB PRO Mk2 generating messages other than those which a DMX USB Pro1 would generate.

By default, apart from two new messages (Set API Key and Query Hardware Version), the USB PRO Mk2 supports the same application messages as the USB Pro1. To enable support for all messages in this API document, an API key must be used to unlock the extended features of the USB PRO Mk2.

To detect if a DMX USB Pro1 or a USB PRO Mk2 is present, the Query Hardware Version message should be used. The DMX USB Pro1 will not reply to the Query Hardware Version request.

Please keep in mind, the Hardware Version value may change in the future to indicate a new revision of USB PRO Mk2.

Application Messages

1. Get Port Widget Parameters Request (Label=3 for port 1, Label=228 for port 2)

This message requests the non-volatile Widget configuration for one DMX port.

Size In Bytes	Description
1	LSB of user configuration size in bytes. Valid range for user configuration size is 0 to 508.
1	MSB of user configuration size in bytes.

2. Get Port Widget Parameters Reply (Label=3 for port 1, Label=228 for port 2)

The Widget sends this message to the PC in response to the Get Port Widget Parameters request.

Size In Bytes	Description
1	Firmware version LSB. Valid range is 0 to 255.
1	Firmware version MSB. Valid range is 0 to 255.
1	DMX output break time in 10.67 microsecond units. Valid range is 9 to 127.
1	DMX output Mark After Break time in 10.67 microsecond units. Valid range is 1 to 127.
1	DMX output rate in packets per second. Valid range is 1 to 40, or 0 for fastest rate possible.
user_configuration_size	User defined configuration data. See Set Port Widget Parameters request.

3. Set Port Widget Parameters Request (Label=4 for port 1, Label=202 for port 2)

This message sets the non-volatile Widget configuration for one DMX port, which is preserved when the Widget loses power.

Size In Bytes	Description
1	LSB of user configuration size in bytes. Valid range for user configuration size is 0 to 508.
1	MSB of user configuration size in bytes.
1	DMX output break time in 10.67 microsecond units. Valid range is 9 to 127.
1	DMX output Mark After Break time in 10.67 microsecond units. Valid range is 1 to 127.
1	DMX output rate in packets per second. Valid range is 1 to 40, or 0 for fastest rate possible (this will make the most difference when the output universe size is smallest).
user_configuration_size	User defined configuration data.

4. Received DMX Packet (Label=5 for port 1, Label=185 for port 2)

The Widget sends this message to the PC unsolicited, whenever the Widget receives a DMX or RDM packet from a DMX port, and the Receive DMX on Change mode for that DMX port is 'Send always'.

Size In Bytes	Description
1	DMX receive status. When this is 0, the DMX data in this message is valid. When this is nonzero, the DMX data in this message is

Size In Bytes	Description
	corrupted. Bit 0: 0=No error,1=Widget receive queue overflowed. Bit 1: 0=No error,1=Widget receive overrun occurred.
1 to 513	Received DMX data beginning with the start code. The size of the received DMX data can be determined from the overall message size.

5. Output Only Send DMX Packet Request (Label=6 for port 1, Label=134 for port 2)

This message requests the Widget to periodically send a DMX packet out of one of the Widget DMX ports at the configured DMX output rate for that DMX port. This message causes the Widget to leave the DMX port direction as output after each DMX packet is sent, so no DMX packets will be received as a result of this request.

If a show is playing on the input port, this request will stop the show playing.

The request will be ignored if the port is disabled (see Set Port Assignment request).

Size In Bytes	Description
25 to 513	DMX data to send, beginning with the start code. The overall message size specifies the size of the DMX data to send, and also sets the universe size (the number of DMX channels which are output).

The periodic DMX packet output will continue until the Widget receives any request message that changes the DMX port direction to input. The request messages that change the DMX port direction to input are:

Receive DMX ON Change Request,
Reprogram Firmware Request,
Send RDM Discovery Request if no show is playing on the port in the request,
Send RDM Packet Request if no show is playing on the port in the request.

6. Send RDM Packet Request

(Label=7 for port 1, Label=137 for port 2)

This message requests the Widget to send an RDM packet out of one of the Widget DMX ports, and then change the DMX port direction to input, so that RDM or DMX packets can be received.

If a show is playing on the input port, the show will resume playing after this request.

The request will be ignored if the port is disabled (see Set Port Assignment request).

Size In Bytes	Description
1 to 513	RDM data to send, beginning with the start code. The overall message size specifies the size of the RDM data to send.

7. Receive DMX ON Change (Label = 8 for port 1, Label=223 for port 2)

This message requests the Widget to send a DMX packet to the PC only when the DMX values change on the input DMX port.

By default the widget will always send the whole DMX packet, if you want to send on change it must be enabled by sending this message.

This message also reinitializes the DMX receive processing, so that if change of state reception is selected, the initial received DMX data is cleared to all zeros.

Size In Bytes	Description
1	0: Receive every DMX/RDM packet 1: Receive DMX changes only. RDM packets can NOT be received in this mode.

8. Received DMX Change Of State Packet (Label=9 for port 1, Label=146 for port 2)

The Widget sends one or more instances of this message to the PC unsolicited, whenever the Widget receives a changed DMX packet from a DMX port, and the Receive DMX on Change mode for that DMX port is 'Send on data change only'.

Size In Bytes	Description
1	Start changed byte number.
5	Changed bit array, where array bit 0 is bit 0 of first byte and array bit 39 is bit 7 of last byte.
1 to 40	Changed DMX data byte array. One byte is present for each set bit in the Changed bit array.

The user program can decode the message into a 513 byte received DMX data array, beginning with the start code. The algorithm to do this is shown below:

```
On startup, zero out the 513 byte received_dmx_array
For each Change Of State packet received
   changed_byte_index = 0
   For bit_array_index = 0 to 39
      If changed_bit_array[bit_array_index] is 1 then
         received_dmx_array[start_changed_byte_number * 8 + bit_array_index] =
            changed_dmx_data_array[changed_byte_index]
         Increment changed_byte_index
      Endif
   Endfor
Endfor
```

9. Get Widget Serial Number Request (Label = 10, no data)

This message requests the Widget serial number, which should be the same as that printed on the Widget case.

10. Get Widget Serial Number Reply (Label = 10)

The Widget sends this message to the PC in response to the Get Widget Serial Number request.

Size In Bytes	Description
4	BCD serial number, with LSB stored at lowest address.

11. Send RDM Discovery Request (Label=11 for port 1, Label=157 for port 2)

This message requests the Widget to send an RDM Discovery Request packet out of one of the Widget DMX ports, and then receive an RDM Discovery Response (see Received DMX Packet).

If a show is playing on the input port, the show will resume playing after this request.

The request will be ignored if the port is disabled (see Set Port Assignment request).

Size In Bytes	Description
38	DISC_UNIQUE_BRANCH RDM request packet to send.

12. RDM Controller Receive Timeout (Label=12 for port 1, Label=214 for port 2, no data)

The RDM controller received no reply message within receive timeout time (This message was added in DMX USB Pro1 firmware v2.4).

This message will only be generated when a Widget DMX port is being used as an RDM controller.

The timeout message will follow the **RDM discovery** reply message, whether or not the reply is partial or complete.

The timeout message will follow the RDM reply message (GET or SET), only when the reply is incomplete or unrecognizable.

13. Set API Key Request (Label = 13)

This message enables either the DMX USB Pro1 API (API1) messages or the USB PRO Mk2 API (API2) messages.

Labels in the range 1 to 12 inclusive, belong to API1. All labels in this document belong to API2.

When the USB PRO Mk2 is powered on, only the API1 messages are enabled. To enable API2 messages a magic number must (**BCC47CA1**) be sent.

Each time the USB PRO Mk2 receives any message which is not enabled, the message will be discarded and the Red LED will blink to show that the message was rejected.

Size In Bytes	Description
4	API key, with LSB stored at lowest address. The API key value to enable API1 is hex 0. The API key value to enable API2 is hex **BCC47CA1**.

LSB Storage means the last byte of the key is sent FIRST and the first bytes sent LAST

14. Query Hardware Version Request (Label = 14, no data)

This message requests the hardware version number of the USB PRO Mk2. The request is supported whether or not the USB PRO Mk2 firmware is valid.

The DMX USB Pro1 will not reply to this request, since the DMX USB Pro1 does not support the request.

15. Query Hardware Version Reply (Label = 14)

The Widget sends this message to the PC in response to the Query Hardware Version request.

Size In Bytes	Description
1	Hardware version. The following are valid values for USB PRO MK2 2: DMX USB PRO MK2 3: DMX USB PRO MK2 revB

16. Get Port Assignment Request (Label =233, no data)

This message requests the DMX/MIDI port assignment.

17. Get Port Assignment Reply (Label =233)

The Widget sends this message to the PC in response to the Get Port Assignment request.

Size In Bytes	Description
1	DMX port 1 assignment. 0: DMX port disabled.

Size In Bytes	Description
	1: DMX port enabled for DMX and RDM.
1	DMX port 2, MIDI IN port and MIDI OUT port assignment. 0: Ports disabled. 1: DMX port enabled for DMX and RDM. 2: MIDI IN and MIDI OUT ports enabled.

When using the MIDI IN port, the DMX port 2 should not be driven, otherwise the DMX signal could corrupt the MIDI IN data.

18. Set Port Assignment Request (Label =182)

This message sets the Widget DMX port assignment, which is not preserved when the Widget loses power.

Size In Bytes	Description
1	DMX port 1 assignment. 0: Port disabled. 1: Port enabled for DMX and RDM.
1	DMX port 2, MIDI IN port and MIDI OUT port assignment. 0: Ports disabled. 1: DMX port enabled for DMX and RDM. 2: MIDI IN and MIDI OUT ports enabled.

19. Received MIDI (Label =138)

The Widget sends this message to the PC unsolicited, whenever the Widget receives data on the MIDI IN port, and the MIDI IN port is enabled.

Size In Bytes	Description
1 to 512 (normally 1)	Received MIDI data bytes.

20. Send MIDI Request (Label =131)

This message requests the Widget to send MIDI data out of the MIDI OUT port.

Size In Bytes	Description
1 to 64	MIDI data to send. The overall message size specifies the size of the MIDI data to send.

21. Show Query Request (Label =221, no data)

This message requests the size of the non-volatile show storage memory.

22. Show Query Reply (Label =221)

The Widget sends this message to the PC in response to the Show Query request.

Size In Bytes	Description
4	Size of the non-volatile show storage memory in bytes.
1	Memory busy status. 0: Erase or write has completed. 1: Erase or write is in progress.

23. Show Block Erase Request (Label =159)

This message erases one block of the non-volatile show storage memory, so that all bytes are set to hex FF. The block size is 64 kbytes for all blocks, except for the first block which may be smaller. The show storage memory must be erased once before a new show can be written to the show storage memory.

After the PC issues a show erase request, the PC should issue show query requests, to detect when the erase has finished. This is necessary to prevent the PC issuing another show erase request or show write request while the previous erase or write request is still in progress, which could cause loss of API messages.

Size In Bytes	Description
4	Subcommand character array, set to 'ERAS'.
2	Block number to erase. Valid range is 0 to (number of show memory blocks – 1), where the number of show memory blocks is the integer (size from show query reply + hex FFFF) / hex 10000.

24. Show Sector Erase Request (Label =159)

This message erases one sector of the non-volatile show storage memory, so that all bytes are set to hex FF. The sector size is 4 kbytes. The show storage memory must be erased once before a new show can be written to the show storage memory.

After the PC issues a show erase request, the PC should issue show query requests, to detect when the erase has finished. This is necessary to prevent the PC issuing another show erase request or show write request while the previous erase or write request is still in progress, which could cause loss of API messages.

Size In Bytes	Description
4	Subcommand character array, set to 'ERSE'.
2	Sector number to erase. Valid range is 0 to (number of show memory sectors – 1), where the number of show memory sectors is the integer (size from show query reply) / hex 1000.

25. Show Write Request (Label =159)

This message writes data to the non-volatile show storage memory. The show storage memory must be erased once before a new show can be written to the show storage memory.

After the PC issues a show write request, the PC should issue show query requests, to detect when the write has finished. This is necessary to prevent the PC issuing another show erase request or show write request while the previous erase or write request is still in progress, which could cause loss of API messages.

See Show File Format section for a description of the show format.

Size In Bytes	Description
4	Subcommand character array, set to 'WRIT'.
4	Address. Valid range is 0 to (size from show query reply – number of data bytes to write).
1 to 256	Data to write.

26. Show Read Request (Label =232)

This message reads data from the non-volatile show storage memory.

Size In Bytes	Description
4	Address. Valid range is 0 to (size from show query reply – number of data bytes to read).
2	Number of bytes to read, with LSB stored at low address. Valid range is 1 to 256.

27. Show Read Reply (Label =232)

The Widget sends this message to the PC in response to the Show Read request. See Show File Format section for a description of the show format.

Size In Bytes	Description
4	Address from show read request.
1 to 256	Data read.

28. Start Show Request (Label =159)

This message starts replay of the stored show if the CRC of the stored show header record is good.

Size In Bytes	Description
4	Subcommand character array, set to 'STAR'.

29. Stop Show Request (Label =159)

This message stops replay of the stored show.

Size In Bytes	Description
4	Subcommand character array, set to 'STOP'.

RDM Implementation Details

Enttec has an RDM protocol PC library that works with the Widget, for creating RDM capable PC applications.

The RDM protocol is a half duplex request/response protocol. To comply with the RDM protocol, the RDM Responder must send only a single response message in reply to each request message from the RDM Controller.

The two RDM ports are considered to be separate RDM devices, as each RDM port is assigned a different UID. Each UID consists of the Enttec manufacturer ID concatenated with a number derived from the Widget Serial Number, as follows:

UID1[0]=hex 45, UID1[1]=hex 4E, UID1[2]=N1[3], UID1[3]=N1[2],
UID1[4]=N1[1], UID1[5]=N1[0].

UID2[0]=hex 45, UID2[1]=hex 4E, UID2[2]=N2[3], UID2[3]=N2[2],
UID2[4]=N2[1], UID2[5]=N2[0].

where
N1 = 2 * SERIAL_NUMBER,
N2 = N1 + 1.

There is a special message with its own label for sending the RDM Discovery request, to enable the Widget to receive the discovery response, which has no break unlike all other types of RDM packet. The Send RDM request should be used to send any RDM packet other than the RDM discovery request.

The behaviour of the RDM Responder Widget is specified in the following table.

Request Received By RDM Responder Widget	**Reply Sent By RDM Responder Widget**	**Request Message Passed To PC**
CC=DISCOVERY_COMMAND PID=DISC_UNIQUE_BRANCH	Discovery response when Widget UID is inside range of UIDs in request message and RDM discovery has not been muted.	No
CC=DISCOVERY_COMMAND PID=DISC_MUTE	DISC_MUTE response when request message was not broadcast.	No
CC=DISCOVERY_COMMAND	DISC_UN_MUTE response when request message was	No

Request Received By RDM Responder Widget	**Reply Sent By RDM Responder Widget**	**Request Message Passed To PC**
PID=DISC_UN_MUTE	not broadcast.	
CC=GET_COMMAND or CC=SET_COMMAND	ACK_TIMER response with time of 0.1 seconds.	Yes. PC should respond with Queued message.
CC=GET_COMMAND PID=QUEUED_MESSAGE	Queued message (see ACK_TIMER).	No
Any unrecognised message or message with bad RDM checksum.	None	Yes

The RDM Responder Widget has space to store a single queued RDM message only, per DMX port. The ACK_TIMER and queued message mechanism ensure that the timing requirements of the RDM specification can be met, even though the PC application is slow to respond due to USB latency or scheduling delays for example.

Show File Format

This section specifies the format of the single show file that can be stored and replayed by the USB PRO Mk2. The show file consists of the following consecutive sections:

Header record.
Frame records.

1. Header Record

Size In Bytes	Description
4	File format version, set to 'SFA1'.
128	Show name text to identify the file. This field is not used by the USB Pro2.
4	Number of frame records, with LSB stored at low address. Minimum valid value is 1.
2	Number of DMX slots per frame record. Valid range is 1 to 512.
1	Bit 0: DMX port to play show on, where 0=DMX1, 1=DMX2. Bit 1: Play show at power on, where 0=disabled, 1=enabled. Bit 2: Backup enable, where 0=disabled, 1=enabled. When backup is enabled, DMX1 will pass through to DMX2, and loss of DMX on DMX1 will start the show on DMX2. Show is invalid if Backup enable is set and DMX port is DMX1.
1	Delay before looping, in seconds. Valid range is 0 to 255.
2	Loop count, with LSB stored at low address. Valid range is 0 to 65534. The show will play (Loop count + 1) times. 65535 is loop forever.
112	Unused, set to hex FF.
2	CRC calculated over rest of header record, with LSB stored at low address.

The CRC is calculated using the XMODEM CRC polynomial (hex 1021), and initial CRC value of 0.

The CRC is used to detect corruption of the record. If the CRC is bad, the show will not play.

2. Frame Record

Each frame record specifies when to play a DMX frame in the show, as well as the DMX data to output for one scene of the show. The number of DMX slots output, excluding the DMX start code, is specified in the header record.

Size In Bytes	**Description**
4	Time in milliseconds, with LSB stored at low address.
1 to 512	DMX data bytes.
2	CRC calculated over rest of frame record, with LSB stored at low address.

The CRC is calculated using the XMODEM CRC polynomial (hex 1021), and initial CRC value of 0.

The CRC is used to detect corruption of the record. If the CRC is bad, the show will stop playing when the bad frame record is read.