OPERATING MANUAL- SECTI0N 2
FOR

DMXter4/4A™ RDM

Software V 4.37
Advanced features manual

Advanced RDM Controller, RDM Sniffer
Scope Trigger, Colortran Support

Goddard Design Co.
51 Nassau Avenue
Brooklyn, NY 11222
(718)599-0170
(718)599-0172 fax
http://www.goddarddesign.com
sales@goddarddesign.com

 01/11/2019
Y:\word_p\wp6doc\dmx-man\V437\dmxman4.37_sec2.wpd

This page left blank

TABLE OF CONTENTS

* * * *

16. ADVANCED RDM

These features are only available if you have the developer's package installed.

The Advanced RDM menu along with the View Response Timing menu are designed primarily for people who are designing, or verifying RDM responders. The DMXter4 saves data on the last 200 packets sent or received. The result code of the packet including any errors detected is saved. Further, a detailed capture criteria can be specified (16.2) Any packet that meets the capture criteria is so marked in the saved data. The last raw captured response and request packet are also saved. See 16.1.6, 16.1.7, & 16.6.3. The last RDM request and response packets are saved as well. See 16.1.8, 16.1.9, & 16.6.4.

Advanced RDM also gives the user the ability to construct and save custom RDM packets. Any legal or illegal packet can be quickly constructed. These packets can then be sent to a responder once or repeatedly.

16.1 Browse Packet History

The browse menu allows scrolling thru recent RDM packets. The following keys are active:
<UP> and **<DOWN>** move you thru browse data.
<LEFT> and **<RIGHT>** shift you in a circular fashion thru the three displays needed to show all the information.
The combination of **<RIGHT><UP>** changes the display of some of the fields from decimal to hex and back.

Each record is for a pair of packets, the RDM request sent by the DMXter4 and the RDM response (if any) from the controller.

16.1.1 Browse Display One

```
|xxx G DEVICE INFO    |
|CA GOOD RESPONSE     |
|
```

1) xxx = the Transaction Number (TN) from the request packet. This rolls over every 256 packets.
2) G = GET, S= SET, D =Discovery
3)DEVICE INFO = A descriptive name for the Parameter ID
4) CA ='Captured', this packet met the capture requirements set in section 16.2. If it was the last packet to meet these requirements, the raw packet may be viewed by either items 16.1.6, 16.1.7 & 16.6.3.

If the controller sent a get for the PID "Get Queued Messages", the display will be slightly different. Queued messages are unique in RDM in that one requesting PID may trigger the return of any response PID that the responder sees fit to send. It could be the slot address of that responder, letting the controller know that the address has been manually changed. Then the display would be as shown below.

```
|xxx G QUEUED MSGS    |
|CA    g DMX START ADDR|
```

16.1.2 Browse Display Two

```
|CC  PID   PDL  TIME |
|21h 0060h 13h 426.29|
```

This data is for the response.

1) CC = 21 = Command Class Response	displayed in decimal or hex (hex shown)
2) PID = Parameter ID	displayed in hex only
3) PDL= Parameter Data Length	displayed in decimal or hex (hex shown)

16.S Shortcut Menu

The <RED> key brings up a second menu that provides quicker access to menu items that you may need more often.

The list below is for units fitted with Advanced RDM option.

```
1 BROWSE PKT HISTORY
2 VIEW CAPT REQ RAW
3 VIEW CAPT RESP RAW
4 VIEW PREV REQ RAW
5 VIEW PREV RESP RAW
6 RESET RESP. TIMING
7 EDIT NULL SC DATA
8 SEND INFO TO USB
9 BACK TO RDM MENU
```

4) Time = A time stamp in seconds since the last time the top key was pressed. Note that "Time" rolls over every 655.36 seconds (approximately 11 minutes)
This display is for the response to a DMXter request.

16.1.3 Browse Display Three

```
|RQ DLY BK MAB IST LN|   |RQ DLY BK MAB IST LN|
|   L H LH L H L H LH|   | Q   H    L        H|
```

Only the oldest packet in the browse memory will have a display with most of its flag fields filled in.
This display highlights information about the packet.
(R) An R in the bottom row means that the **next** packet is a retry of this packet.
(Q) A Q in the bottom row means that the **next** packet will be a get Queued message.
The other fields in this display highlight whenever a new maximum (H) or new minimum (L) timing for a packet was seen. The first packet after clearing the browse memory is by definition the slowest and fastest packet at the same time. The fields in order are
DLY - preceding interpacket delay,
BK - break,
MAB - mark before break,
IST - inter slot time,
LN - total packet length.

16.1.4 Browse Display Four

```
|REQ:CC  PID   PDL    |
|    20h 0060h 00h    |
```

This display is a summary of the request packet.

16.1.5 Display of Discovery & Broadcast messages

A discovery message that gets no reply is shown below. There are no devices on this branch of the tree.

```
|99h D DISC UNIQ BRA |   |CC  PID   PDL  TIME |
|  LEVEL   3 IDLE    |   |(EMPTY)        20.70|

|RQ DLY BK MAB IST LN|   | REQ:CC   PID   PDL |
|     H    L        H|   |      10h 0001h 0Ch |
```

A discovery message that got one or more replies is shown below. The response may still be just junk and we do not attempt to display it here. The raw data can be captured.

```
|A1h D DISC UNIQ BRA |   |CC  PID   PDL  TIME |
|CA LEVEL  48 ACTIVITY|  |(EMPTY)        16.70|

|RQ DLY BK MAB IST LN|   | REQ:CC   PID   PDL |
|     H    L        H|   |      10h 0001h 0Ch |
```

Broadcast and Vendor-cast messages do not get a response, so only the outgoing message is displayed.

```
|A5h S IDENTIFY DEV  |   |CC  PID   PDL  TIME |
| BROADCAST          |   |(EMPTY)        26.70 |

|RQ DLY BK MAB IST LN|   | REQ:CC  PID   PDL  |
|     H    L        H|   |     30h 0060h   0  |
```

16.1.6 View Raw Captured Request Packet

Do we need to say more? See below for display details

16.1.7 View Raw Captured Response Packet

16.1.8 View Raw Previous Request Packet

The last packet sent is stored and may be inspected by this menu item. See below for display details

16.1.9 View Raw Previous Response Packet

The last packet received from a responder is stored and may be inspected by this menu item. See below for display details. All of the above are items that let you view a raw packet generated by either the DMXter4 controller or the DUT responder. All of these items can switch from showing slot number to showing slot description as shown below.

```
      |SLT:   1   2   3   4|   |   SUB STARTCODE    |
      |RDM:  1h 28h 47h 44h|   |RDM:  1h 28h 47h 44h|

|SLT:   3   4   5   6|   |     DEST. UID5     |   |SLT:   3   4   5   6|
|RDM: 47h 44h 00h 40h|   |RDM:  'G' 'D'' ' '@'|   |RDM:  'G' 'D'' ' '@'|
```

Pressing <UP><DOWN> switches between a numeric top line and a description of the data in the left most slot. Using the <RIGHT><UP> combination cycles you thru decimal, hex and **ASCII** data formats.

One other addition to RDM packet displays is that you can now view the Start code slot, aka slot zero. This display is a bit jarring for old DMXter users, but in this mode slot 0 falls between slot 512 and slot 1. See below.

```
      |SLT:   0   1   2   3|
      |RDM: CCh  0h 28h 47h|
```

16.2 Setting Packet Capture Criteria

```
      | CAPTURE PACKET ON  |
      |▸ALL◂OK ERR CORRUPT |
```

The packet capture routine is flexible and hence a bit dense. First we need to decide if we want to capture 'ALL' packets that meet our other capture requirements or just the ones that are 'OK', the ones that have transmission 'ERRors', or the ones that have a CORRUPT format. You also have the choice of selecting both packets that are corrupt and have errors.

What we mean by capture needs to be explained. All RDM packets that are sent or received have a list of information stored about them in the packet history. Only the last 200 packets' data can be viewed. Any packet that meets the capture requirements is marked in the history and the full packet is stored temporarily. However you will only be able to view the last request and response packet that was marked as 'captured'.

After you have made this choice you can set up 8 identical slot criteria displays.

```
      |  SLOT CRITERION 1  |
      |RESP SLOT   0 - ANY |
```

This display allows the selection of a value for any particular slot in an RDM packet. All 8 criteria will be combined in a logical *and* fashion.

First the type of packet can be selected for either a 'RESPonse' or 'REQuest .
Then the slot number for which you want to set a capture value is entered.

```
      |     START CODE     |   | TRANSACTION NUMBER |
      |RESP SLOT   0 - ANY |   |RESP SLOT   15 - ANY|
```

The top line is re-labeled as you change numeric value. Note the slot number is always displayed in decimal. The slot value may be displayed in either hex or decimal. The standard short cut key to change format works here.

```
      |  SLOT CRITERION 1  |   |  SLOT CRITERION 6  |
      |RESP SLOT   0 = CCh |   |RESP SLOT   15 >  45|
```

Note that you can set match requirements to =,<,>,or ≠. Any capture criterion not needed is left in its blank state. After setting the slot criterion you can set the packet length required.

```
      |LEN CRIT (INC SC&CS)|   |LEN CRIT (INC SC&CS)|
      |RESP   LENGTH - ANY |   |REQ.   LENGTH >  25 |
```

The second display above is set for any request packet longer than 25 slots in length.

A final option allows for either an 8 way *and* of the criteria or two, 4 way *ands* that are then *ored* together.

```
|CAPTURE ON SLOT CRIT|
|  >1-8< 1-4 OR 5-8  |
```

The last item lets you reset all of the capture criteria to their clear state.

16.3 Build a Custom Request Packet

The following 4 cursor menus let you build a special packet by hand. We are not saying it's painless, but it's easier than starting to write custom code. . . On entry, the packet displayed will be the last packet sent by the DMXter4 controller. So to modify a packet the DMXter4 already supports, send it. Then go to this menu.

The first cursor menu lets you set

CC	Command Class
PID	Parameter Identifier (when a valid number is entered, it will show a description of the PID)
PDL	Parameter Data Length (set the data field length)
TYPE	packet type
	Your choices are unicast, broadcast, vendor-cast, discovery (DUB)

A few sample displays are shown below.

```
|CC  PID   PDL TYPE  ||PID: CLEAR STATUS ID||PARAMETER DATA LEN. |
|30h 0032h 00h UNICST||30h 0032h 00h UNICST||30h 0032h 00h UNICST|
```

Pressing <YES/Q> and then <DOWN> to see the next menu.

```
|TN   CS   LEN TXLEN |
|AUTO AUTO 24    26 |
```

The above fields are all filled automatically, but can be edited to create non standard packets. The items on this display are:

TN	Transaction Number
CS	Check Sum
LEN	Active Packet Length
TXLEN	Transmission Length (including the Start Code and Check Sum)

The next display allows you to set some items that are seldom changed

```
|SSC PORT MSG      |
|01H 00h 00h       |
```

Here you can set the Sub Start Code, the Port number and the Message Count.

16.3.1 Edit Raw Request Data

One major item hasn't been set yet, as well as a bunch of items that are rarely sent, at least by a controller. The major item is the content of the PD (Parameter Data). So the next item lets you edit the raw packet for absolute control. To ease editing the display points at the beginning of the PD on entry. While editing raw data, holding **<YES/Q>+<DOWN>** clears the PD field.

```
|SLT:  24  25  26   0|
|RDM: 02h 03h 2Fh CCh|
```

16.3.2 Sending the Custom Packet

This and the next item let you send the packet you just built and see what comes back. On a good reply you will see the PLD of the response and up to four bytes of reply data.

```
|GOOD RESP.  PLD =  2|
|00H 04h             |
```

16.3.3 Viewing the response.

This lets you view the complete returned packet.

16.3.4 Send Packet Repeatedly This routine sends the last packet edited by the build routine continuously. If a custom packet was not built, the last packet sent by the DMXter4/4A will still be in the buffer, and will be sent. By this method, any packet may be sent repeatedly.

The top line of each display counts the number of packets sent and whether the routine is sending or paused. Either the **<LEFT>** or **<RIGHT>** keys will cause the display to switch in and out of the paused mode.

```
|PKT: 1234 SENDING   |  |PKT: 1234 PAUSED    |
|COR:    0 TOUT:    0|  |COR:    0 TOUT:    0|
```

The first display shows how many CORrupt packets, and how many Timed OUT responses were seen.
<UP> and **<DOWN>** will cycle you through the following displays.

```
|PKT: 1234 SENDING   |
|NAK:    0 ACKT:    0|
```

The next display shows how many packets were NAKed, and how many packets were ACK Timered.

```
|PKT: 1234 SENDING   |
|ACK:      ACKO:     |
```

This display shows the number of packets that were ACKed, and how many got an ACK OVERFLOW response.

```
|PKT: 1234 SENDING   |
|      Captured: 1235|
```

This display shows how may packets met the capture requirements.

```
|PKT: 1234 SENDING   |
|PDL Range  16 to  16|
```

What was the shortest and longest Parameter Data Length seen?

```
|PKT:      SENDING  ▵|
|GET   DIVICE INFO  ▿|
```

Servicing the display takes a certain amount of time. If you wish to have the maximum update rate possible, you can use this last static display. This lessens the pause before an RDM request is sent. The update rate will be affected by the turn around time of the responder.

16.3.5 Load Captured Packet
Loads the last packet to meet the capture criteria back into the RDM buffer so it may be sent or edited.

16.3.6 Load Packet from Library
The library can store 25 named custom packets,. This routine is used to load stored packets so they may be sent as custom PIDs.

16.3.7 Store Packets in Library
This allows the storing of named packets. The name is entered by the text editor (See section24)

16.3.8 Delete Library Packet
I think we don't need to explain this one further.

16.4 Specific Tests
This is a group of tests that sends defective packets to check handling of RDM responders. In general these packets should cause no harm; the responder should not crash. Most of these packets should be ignored or NACKed harmlessly.

16.4.1 Maximum Packet Length
This test sends a Get Device Info PID with a PDL of 231. This packet should have a PDL of 0. It should be NACKed, and should not crash the responder. It is the maximum length packet that should ever be sent in RDM. We use the Device Info PID because it's a PID that all responders must support.

16.4.2 Invalid Sub-device
This test sends a Get Device Info PID to sub device 600. Sub devices are numbered from 1 to 512.

This should be NACKed, ‘SUB DEV RANGE'. We use Device Info because it's a PID that all responders must support.

16.4.3 Send GET as a Sub-device All Call

This test sends a GET packet to the Sub-Device All Call Address. The responder should NACK the request with "SUB_DEVICE_OUT_OF_RANGE" per section 9.2.2 of E1.20.

16.4.4 Test for Any Reply to Broadcast Commands

RDM responders should never respond to a message sent to the broadcast address. However, it is an easy error to make when coding. It is unlikely that a responder replies to all broadcast messages, but one class of message may slip through early code testing. We send 6 types of packets.

1) Discovery Mute
2) Get Device Information
3) Set Device Information
4) Set Identify off
5) Get PID 6123h (PLASA Undefined)
6) Set PID 6123h

The last two PIDs make sure that the unknown PID handler doesn't also respond to a broadcast message.

Each PID is sent twice, first to the broadcast address, and then to the vendorcast address. Hopefully you will see a message telling you that nothing responded. If not, you will be told what failed.

```
|0001:DISC MUTE      |         |0070:DEVICE_INFO    |
|RESPONSE TO BROADCST|         |RESPONSE TO BROADCST|
```

Or similar

16.4.5 Length/PDL Mismatch

This test sends a packet where the RDM "Message Length" field does not match the PDL. The RDM checksum is weak so this is another way to check packet validity. The responder should treat this like a checksum error and ignore the request, although NACK'ing is probably also acceptable. Again we use Device Info because it's a PID that all responders must support.

16.4.6 Short Packets

These three tests send a packet that is truncated, missing one or more bytes. They should produce checksum errors. We use Device Info because it's a PID that all responders must support.

1 Byte cutoff

The first test cuts off 1 byte. It should look like a checksum error This packet should be ignored by the responder causing a time out.

16.4.7 Send a Packet Short by 3 Bytes

This test sends a packet missing the PDL and the checksum. It should cause a time out.

16.4.8 Send a Packets of 9 Bytes

This test sends a packet ending after the source UID. It should cause a time out error,

16.4.9 Send a Packet Short by 24 Bytes (2 Byte Runt)

Sends a packet containing only the Start Code and the Sub-start Code. .

16.4.10 Send 1 extra byte.

This test will most likely be passed. The extra garbage bytes are ignored.

16.4.11 Send 3 Extra Bytes

This test will most likely be passed. The extra garbage bytes are ignored.

16.4.12 Send 255 Extra Bytes

This should cause a time out.

16.4.13 Maximum Length Packets with 3 Bytes of Trailing Data

This test sends a Max length RDM packet (Similar to SEND MAX LENGTH PACKET), but with

additional trailing garbage after that. This is mostly intended to test for buffer overflow errors. It may report a Packet Size error. As always, the most important behavior for the responder is "don't crash".

16.4.14 Max Length Packets with 255 Bytes of Trailing Data

This should cause a time out. The do not crash rule still holds. See above.

16.4.15 DUB Short 1 Byte

This Test sends a DUB request that is missing part of the checksum. It should trigger no reply.

16.4.16 DUB With 3 Extra Bytes

This test sends a DUB request with 3 extra bytes of trailing garbage.

16.4.17 Sent Broadcast Short 1 Byte

This test sends a broadcast request that is missing part of the checksum

16.4.18 Send Broadcast - 3 Extra Bytes

This test sends a broadcast request with 3 extra trailing bytes

16.4.19 Send Checksum Error

This test sends packets with invalid Checksums. It sends two packets, one with the high byte of the Checksum corrupted, the other with the low byte corrupted. The responder should ignore the request. The DMXter should see a time out.

16.4.20 Send Invalid Command Class

This test sends a packet with an Invalid Command Class. Technically this should be ignored and not NACK'd since the standard only allows NACK for GET and SET. In the real world, as long as the responder doesn't crash it's probably acceptable.

16.4.21 Send Non Discovery PID With Discovery CC

This test sends a packet with a Discovery Command Class but a non-discovery PID. Technically, a responder can't NACK a Discovery Command Class PID (E1.20-2010 Table 6-7), so the responder should not respond to this request. Nevertheless, if the responder does NACK the request that's acceptable in most real-world cases.

16.4.22 Send Invalid Sub-Start Code

This test sends a packet with an Invalid Sub-Start Code. The responder must ignore this request (E1.20-2010 Section 6.2.2). If the responder accepts it or responds in any way it will likely cause interoperability problems with a future version of this standard that uses the Sub-Startcode to indicate a different packet format.

16.4.23 Send Port ID of Zero

This test sends a packet with the Port ID set of zero. Technically a Port ID of zero is not valid (E1.20-2010 Section 6.2.7.1), but in the real-world this is a no-consequence mistake.

16.4.24 Non-zero Message-Count Sent

It should not crash the controller.

16.4.25 Fast Broadcast Test Packet

This is the entry point for a routine that sends a fast sequence of broadcast commands. It sends the sequence back-to-back, to find responders that take too long to handle a broadcast packet. The sequence is:

Unicast SET IDENTIFY ON
Wait 500ms
Broadcast SET IDENTIFY ON
Broadcast SET IDENTIFY ON
Broadcast SET IDENTIFY ON
Broadcast SET IDENTIFY OFF <---A single "Ident Off" amidst a flood of other broadcasts
Broadcast UNMUTE
Broadcast UNMUTE

If Identify is ON after this sequence then the responder failed the test. If it passed, the DMXter display reads 'Test Passed'. This test is best conducted with NSC interleave disabled and a fast timing flavor ("RDM" or "MIN")

16.4.26 Framing Error

This test sends the RDM request Device Info (0060) with a slot having a framing error. Stop bit one is sent as a space(0) instead of a mark (1). If the responder properly drops the entire packet the DMXter will report 'Response Timed Out'. If the responder accepts this packet the DMXter will, somewhat counterintuitively, display. . .

```
|       WARNING!       |
|   GOOD RESPONSE      |
```

16.5 Setup Scope Trigger Output

The DMXter4 can generate a pulse **at the end** of RDM packets that meet certain requirements. The trigger is output on Pins 4 and 5 of the DMX output connector. The trigger mode is controlled by the menu shown below. The current mode selection is set off by filled arrows pointing to the appropriate text label. They are shown here by the more than and less than symbol. One mode is off.

```
|SCOPE TRIGGER OUTPUT|
| OFF▸REQ◂RESP CAPT  |
```

>REQ< A trigger is generated during any request packet.
>RESP< A trigger is generated during the period allowed for responses.
>REQ RESP< A trigger is generated during the request packet and response period.
>CAPT<A trigger is generated for any packet that meets the capture requirements discussed above within a few hundred microseconds after the response is complete.

16.6 What type of data and errors do we track in RDM packets?

The DMXter4 uses an RDM engine to send Request packets and receive and verify Response packets. Every received packet is evaluated to make sure that it is properly formed and received error free.. You will be able to scroll thru the packet history and see just what packets and what errors were seen.

RDM provides some error responses which a Responder sends to a Controller to indicate that communication has failed. The responder does this by sending a NACK (Negative Acknowledge). A NACK provides a reason code. Reason codes are also displayed when you browse the packet history. The NACK Reason Code is contained in the NACK packet and tells the controller why a packet was NACK'd. The possible codes are given in E1.20 Table A -17.

The table below lists the descriptions saved and displayed by the Browse Packet History menu. The first column is a reference number and is not saved or displayed.

Table 16.6

Item	Reported text for packet history display	ER	CR	NAC	NR	INF	COMMENT
1	CHECKSUM ERROR		X				Likely com link problems
2	FRAMING ERR SB 1	X					Likely com link problems
3	FRAMING ERR SB 2	X					Likely com link problems
4	BAD STARTCODE		X				Likely com link problems or responder failure
5	BAD SUB-STARTCODE		X				The failure of this test means that we connected to different version of the standard.
6	WRONG PDL FOR PID		X				The received packet has a longer PLD block than is allowed for this PID

7	BAD PDL FOR RESPTY		X				ACK_TIMER & NACK_REASON response packets have a fixed length that is not the same as would expected from a normal response to the requesting PID
8	PKT TOO SHORT		X				The packet is shorter than minimum for any RDM packet
9	PKT TOO LONG		X				The packet is longer than allowed for any RDM packet
10	LEN/rxCNT MISMATCH		X				The physical length does match the Message Length field (slot 2)
11	LEN/PDL MISMATCH		X				The Message Length field and the Message Count field are not consistent
12	TN MISMATCH	X					The Transaction Number is not as expected, a packet may have been dropped
13	BAD RESPTYPE		X				There are only 4 allowed response types this packet is trying to invent a new one
14	GOOD RESPONSE					X	Good Packet ready for use!
15	GOT ACK_TIMER					X	TYPE_ACK_TIMER indicates that the responder is unable to supply the requested GET information or SET confirmation within the required response time.
16	GOT ACK_OVERFLOW					X	The responder has more information for the controller than will fit in a single response packet.
17	RESPONSE TIMED OUT		X				No one is at home or they are sound asleep!
18	IDLE LEVEL XXX					X	At discovery level ‘X’ there was no reply
19	CS GOOD LEVEL XXX					X	At discovery level ‘X’ there was 1 reply.
20	CS BAD LEVEL XXX					X	At discovery level ‘X’ there were 1 or more replies The checksum is bad.
21	BROADCAST					X	The controller sent this message to the broadcast address. No reply should be seen.
22	VENDORCAST					X	The controller set this message to all responders of one Man UID. No reply should be seen.
23	NAK:BAD REASN CODE			X			Likely responder error.
24	NACK:UNKNOWN PID			X	0 0		The responder cannot comply with request because the message is not implemented in responder.

25	NACK:FORMAT ERROR			X	01		The responder cannot interpret request as controller data was not formatted correctly.
26	NACK:HARDWAR FAULT			X	02		The responder cannot comply due to an internal hardware fault.
27	NACK:PROXY REJECT			X	03		Proxy is not the RDM line master and cannot comply with message.
28	NACK:WRITE PROTECT			X	04		SET Command normally allowed but being blocked currently.
29	NAK:UNSUP CMDCLASS			X	05		Not valid for Command Class attempted. May be used where GET allowed but SET is not supported.
30	NACK:DATA RANGE			X	06		Value for given Parameter out of allowable range or not supported.
31	NACK:BUFFER FULL			X	07		Buffer or Queue space currently has no free space to store data.
32	NACK:PACKET SIZE			X	08		Incoming message exceeds buffer capacity.
33	NACK:SUB DEV RANGE			X	09		Sub-Device is out of range or unknown
34	NACK REASON 000Ah			X	A		The proxy buffer is full and can not store any more Queued Message or Status Message responses.
35	NACK REASON 000Bh			X	B		reserved
36	NACK REASON 000Ch			X	C		reserved
37	NACK REASON 000Dh			X	D		reserved
38	NACK REASON 000Eh			X	E		reserved
39	NACK REASON 000Fh			X	F		reserved
40	DEST. UID MISMATCH		X				Destination UID is not controllers UID
41	SRC UID MISMATCH		X				Source UID is not the UID of device that was last polled
42	SUBDEVICE MISMATCH						
43	CMD CLASS MISMATCH						
44	PARAM ID MISMATCH						
45	GET QM HAD QM RESP						'Get Queued message had Queued message Response
46	NOISE :Data received but No break						There is 485 line activity but no break
	Reported text for packet history display	ER	CR	NAC	NR	INF	COMMENT

	ER = Error CR corrupt Packet format	NAC =Responder NACKed the request NR = Nack Reason (RDM) Informative = What type of good packet was received

16.7 Dump Packet Data to USB

16.7.1 Dump Packet History (example 1)

```
PACKET HISTORY
   TIME TN  CC   NAME            RESULT            CC  PID   PDL  CC  PID   PDL  RQ|DLY|BRK|MAB|IST|LEN
*  8.81 00h SET  IDENTIFY DEV    BROADCAST         30h 1000h 01h                   |   |   |   |   |
*  8.88 01h SET  IDENTIFY DEV    GOOD RESPONSE     30h 1000h 01h  31h 1000h 00h    |L H|L H|L H|L H|L H
   8.96 02h GET  DEV MODEL DESCR GOOD RESPONSE     20h 0080h 00h  21h 0080h 14h    |L  |  H|   |  H|  H
   9.82 03h SET  IDENTIFY DEV    BROADCAST         30h 1000h 01h                   |   |   |   |   |
   9.90 04h SET  IDENTIFY DEV    GOOD RESPONSE     30h 1000h 01h  31h 1000h 00h    |  H|  H|   |   |
   9.97 05h GET  DEV MODEL DESCR GOOD RESPONSE     20h 0080h 00h  21h 0080h 14h    |   |   |   |L  |  H
  10.97 06h SET  IDENTIFY DEV    BROADCAST         30h 1000h 01h                   |   |   |   |   |
  11.05 07h GET  DEVICE INFO     GOOD RESPONSE     20h 0060h 00h  21h 0060h 13h    |L  |L  |   |   |
  17.93 08h GET  DEVICE INFO     GOOD RESPONSE     20h 0060h 00h  21h 0060h 13h    |  H|   |L  |   |
  18.00 09h GET  DEV MODEL DESCR GOOD RESPONSE     20h 0080h 00h  21h 0080h 14h    |   |   |   |   |
--AT END OF LIST--
```

The fields in the above are:

* = This packet pair satisfies the capture requirements of the Capture Setup Menu , see section 16.2
Time = A time stamp in seconds since the last time the top key was pressed.
TN = The transaction number. It is set by the controller and rolls over at 256.
CC = A text callout of the Command Class of the request.
Name = The name of the request.
Result = What sort of response or error code did the DMXter generate for this request / response pair. See Table 16.5 in section 16.5.
CC = The numeric Command Class of the request in hex.
PID = The Parameter ID of the request in hex.
PDL = The Parameter Data Length of the request in hex.
CC = The numeric Command Class of the response in hex ,if any.
PID = The Parameter ID of the response in hex, if any.
PDL = The Parameter Data Length of the response in hex, if any.

The next fields mark packets that were automatically generated.
R = This packet is a retry.
Q = This packet is a response to a Queued Message.
The other fields in this display are highlighted whenever a new maximum (H) or new minimum (L) timing or length is seen. The first packet after clearing of the browse memory is by definition the slowest and fastest packet at the same time.
DLY = Preceding interpacket delay,
BK = Break,
MAB = Mark Before Break
IST = Inter Slot Time
LEN = Total packet length.

16.7.2 Dump Packet History (example 2)

```
PACKET HISTORY
   TIME TN  CC   NAME             RESULT            CC  PID   PDL  CC  PID   PDL  RQ|DLY|BRK|MAB|IST|LEN
 120.73 22h GET  DEVICE INFO     GOOD RESPONSE      20h 0060h 00h  21h 0060h 13h    |   |   |   |   |
 120.80 23h GET  DEV MODEL DESCR GOOD RESPONSE      20h 0080h 00h  21h 0080h 14h    |   |   |   |   |
 121.67 24h GET  DEVICE LABEL    GOOD RESPONSE      20h 0082h 00h  21h 0082h 08h    |   |   |   |   |
 122.21 25h GET  MFG LABEL       GOOD RESPONSE      20h 0081h 00h  21h 0081h 10h    |   |   |   |   |
 122.69 26h GET  SW VER LABEL    GOOD RESPONSE      20h 00C0h 00h  21h 00C0h 20h    |   |   |   |   |
 181.17 27h GET  SW VER LABEL    GOOD RESPONSE      20h 00C0h 00h  21h 00C0h 20h    |   |   |   |   |
 183.55 28h GET  DEVICE INFO     GOOD RESPONSE      20h 0060h 00h  21h 0060h 13h    |   |   |   |   |
 183.62 29h GET  DEV MODEL DESCR GOOD RESPONSE      20h 0080h 00h  21h 0080h 14h    |   |   |   |   |
 184.43 2Ah GET  DEVICE LABEL    GOOD RESPONSE      20h 0082h 00h  21h 0082h 08h    |   |   |   |   |
 185.24 2Bh GET  MFG LABEL       GOOD RESPONSE      20h 0081h 00h  21h 0081h 10h    |   |   |   |   |
 185.84 2Ch GET  SW VER LABEL    GOOD RESPONSE      20h 00C0h 00h  21h 00C0h 20h    |   |   |   |   |
 187.45 2Dh GET  DMX PERS DESCR  GOOD RESPONSE      20h 00E1h 01h  21h 00E1h 0Dh    |   |   |   |   |
 195.73 2Eh GET  POWER STATE     GOOD RESPONSE      20h 1010h 00h  21h 1010h 01h    |   |   |   |   |
 196.44 2Fh GET  DEVICE HOURS    GOOD RESPONSE      20h 0400h 00h  21h 0400h 04h    |   |   |   |   |
*196.51 30h GET  DEV PWR CYCLES  GOOD RESPONSE      20h 0405h 00h  21h 0405h 04h    |   |   |   |   |
 197.88 31h GET  LAMP HOURS      GOOD RESPONSE      20h 0401h 00h  21h 0401h 04h    |   |   |   |   |
 197.96 32h GET  LAMP STRIKES    GOOD RESPONSE      20h 0402h 00h  21h 0402h 04h    |   |   |   |   |
 199.66 33h GET  LAMP STATE      GOOD RESPONSE      20h 0403h 00h  21h 0403h 01h    |   |   |   |   |
 199.74 34h GET  LAMP ON MODE    GOOD RESPONSE      20h 0404h 00h  21h 0404h 01h    |   |   |   |   |
--AT END OF LIST--
```

The above packet history dump shows only one packet that satisfies the capture requirements presently set. That is the packet at time stamp.196.51.(bold added) Since it is the only packet pair to trigger capture, the complete packet dump for the request and response are shown below.

16.7.3 Dump Captured Packets

Below is the complete dump of the packet pair from time stamp 196.51. If multiple packets had caused a capture; the last one would be the one displayed. If one wants to capture all the packets in a session one should check the RDM sniffer software for the DMXter4 RDM.

```
CAPTURED PACKET
 SC SS LEN ------DEST------ -------SRC------- TN PR MC SUBDV CC PID- PDL DATA
REQUEST
 CC 01 18 48 45 00 00 02 5A 47 44 00 40 40 03 30 01 00 00 00 20 04 05 00 03 36
RESPONSE
 CC 01 1C 47 44 00 40 40 03 48 45 00 00 02 5A 30 00 00 00 00 21 04 05 04 00 00 00 05 03 43
--AT END OF LIST-
```

16.7.4 Dump Previous Packets

Below is the complete dump of the packet at time stamp 199.74 Whatever packet seen last would be available to dump at this menu item.

PREVIOUS PACKET
SC SS LEN ------DEST------ -------SRC------- TN PR MC SUBDV CC PID- PDL DATA
REQUEST
CC 01 18 48 45 00 00 02 5A 47 44 00 40 40 03 34 01 00 00 00 20 04 04 00 03 39
RESPONSE
CC 01 19 47 44 00 40 40 03 48 45 00 00 02 5A 34 00 00 00 00 21 04 04 01 00 03 3B
--AT END OF LIST–

16.7.5 Dump Table of Devices

TABLE OF DEVICES
08 08 87 65 43 21
48 45 00 00 02 37
48 45 00 00 02 3A
48 45 00 00 02 3B
48 45 00 00 02 3C
58 45 12 34 56 78
--AT END OF LIST

16.7.6 Dump Responder Timing

```
RESPONDER TIMING       MIN   PRV   MAX
RESPONSE DELAY IN us   328   367   606
BREAK LENGTH IN us     177   193   198
MAB LENGTH IN us        29    29    29
INTERSLOT TIME IN us     0           1
TOTAL LENGTH IN us    1350  1937  2756
DISC. FIRST ACTIVITY   348   356   694
DISC. LAST ACTIVITY   1342  1395  1428
```

16.8 Autofade Null Start Code

```
|SLOT WAVE RATE LEVEL|  |SLOT WAVE RATE LEVEL|
|   1 _--     1    0%|  | 512 RISE  200   30%|
```

This item allows one of three wave forms to be sent to any slot They are sent in the presence of RDM. Once started this routine will run until the unit leaves the RDM Controller and returns to the Main Menu. The selectable wave forms are: TRI - a triangle wave, FALL- a falling sawtooth wave, and RISE - a rising sawtooth wave . The rate controls how large a step is added to the slot for each Null packet sent. It will take 256 packets to fade all the way up or down with the rate set to 1. At a rate of 256 every packet will be either full or zero, and the opposite of the last packet. The exact fade rate is dependent on the flavor setting and how often RDM packets are sent. On entry a steady state value is sent to the slot. It can be set by manually adjusting the level.

17 RDM Flavors

Yes, we can set different flavors for RDM This menu comes after the Advanced RDM menu and the BACK TO MAIN MENU item at very end of the RDM loop. It is only seen when Advanced RDM is fitted.

```
|TIMING FLAVOR:▸SLOW◂|
| RDM MIN USER: ABC |
```

RDM Controller Flavors						
	Break	MAB	Inteslot Time	MBB	Slots per Packet*	
Slow	200µs	22µs	20µs	198µs	512	
RDM	176µs	14µs	0µs	178µs	512	
MIN	88µs	9µs	0µs	178µs	512	
User A#	160µs	20µs	20µs	16ms	128	user setable
User B#	160µs	20µs	20µs	880 µs	512	user setable
User C#	160µs	20µs	20µs	16ms	512	user setable

= The user setable flavors are set in the standard flavor menu in the transmit section.
* = Slots per packet only affects null packets interleaved with RDM. RDM packets are of the length required for the packet type being sent.

Section	Default	
16.8.1 Lost Packet Timeout	3440µs	Table 3.2 line 5 + the maximum Break and MAB[1]
16.8.2 Slot Timeout	2144µs	Table 3.3 line 1, plus 1 slot time. This added slot time is required by how we measure this timing.
16.8.3 Discovery Timeout	5800µs.[2]	Table 3.2 line 2
16.8.4 Null Packet Interleave	2	

1) The DMXter does not know when a packet has ended until we have seen the next Break/MAB pair It can only determine retroactively if the line activity was a break, a valid byte, or noise. So, after we've received a full break+MAB we go back and evaluate if there was a timeout. Hence the timeout is the 3000us from Table 3.2 line 5, plus the longest possible Break+MAB.
2) This is the minimum that a controller must wait before sending any packet after sending a discovery packet. After this time the controller may consider that discovery response is finished or lost.

******

*** * * ***

18 RDM SNIFFER

The sniffer is designed to monitor transactions on an RDM line. It is used for testing and debugging both controllers and responders. While it is running it stores data and timings for up to 500 RDM transactions on the line. It time stamps every transaction. It does not store the packet data for non RDM packets. It does do timing measurement on them. Some of the data are viewable while being collected; all of the data may be viewed after the fact. Some information will show up in multiple menu items. We hope this makes it easier to view the information you need.

On entry to the sniffer you will be asked if it is 'at End of Line?', Then the fun begins.
There are seven sub menus, eight if you count returning to the calling menu. Briefly they are;

- Start Sniffing -The data collector and timing measurement engine.
- Browse Packet History

- Timing summary
- Match Criteria Setup
- Sniffer options
- Send Information to the USB
- Back to Main Menu
- Clear Packet History

18.1 Menu One - Data Collection

```
|     RDM SNIFFER      |
|  START SNIFFING ?    |
```

This is the activation point for the Sniffer. You will not leave the Sniffer mode unless you hit the <TOP> key or press <YES/Q> in 'Back to Main Menu' item, which is <UP> 2 presses from the 'Start Sniffing' item.

Starting sniffing starts the live data monitor and collector routine. If you start it and let it run you will see a display similar to one of the two below. A count of all packets and a count of the RDM packets are displayed. The last RDM transaction is briefly displayed on the second line of the display.

```
|36542  PK,   14 RDM |   |36542  PK,  250 RDM |
|  (NO RDM TRAFFIC)  |   |  GR  SENSOR VALUE  |
```

The packet counters are all 6 digits and freeze at >999K While the DMXter is monitoring the line, if you press the **<DOWN>** key you will enter a group of packet summary displays. The first one is shown below.

18.2 Live Packet Timing

```
|     RDM REQUESTS     |
|   12   SHOW TIMING?|
```

This is a live display. The number displayed is the number of RDM request packets since the sniffer was reset. There are four additional displays in the same format. They track the number of RDM Broadcast Request packets, Discovery Response packets, Unknown Packets, Null Start Code Packets, and Alternate Start Code Packets. Each of these displays allows you to view the live timings for the class of packet that it counts.

The timing values are also viewable after the sniffer is stopped through the Timing Summary menu below. Timings for EVERY packet are measured and stored, and you will be able to individually view them in the Browse Packet History menus. You will also be able to dump all the data to a PC via the USB port. The timings we report are:

Idle Before	(How long since the end of the last packet)
Break Length	
MAB	
Interslot Time	
Total Length	(of the packet)
Idle after	(This is identical to the idle before of next packet.)
Slots (Inc. Start Code)	(The number of slots in the packet including the Start Code)
Received Count	(The number of RDM packets received vs the number that met the Match Criteria requirements.)

18.2.1 Non Timing Data Collected

The following events have menus and are counted but do not have timing values:
Matching Packet Count, Dropped Packet Count, Double Break Count, and RDM Checksum Errors.
In this submenu we also sneak in the 'Reset Timing?' question.

18.3 Browse Packet History

Ok, you have stopped the sniffer and you wish to look at what you just got. On entry you will be viewing the last RDM request recorded. There are now nine (yes, 9) displays' worth of data available for every packet. **<LEFT>**and **<RIGHT>** scroll you through the displays for a packet. **<UP>** and **<DOWN>** move you from packet to packet. The **<UP><RIGHT>** shortcut will scroll you through display format options.

We hope most of the labeled data is fairly obvious. The first display requires the most explaining. ^{CA} stood for Captured - the name used in the controller software. In the sniffer we capture all the RDM packets but allow you

to set up matching criteria for packets to be marked. [CA] marks those packets you wish to look at. The next unmarked number (C7 in the example) is the Transaction Number (TN) of a pair of request /response packets. It is set by the controller and echoed by the responder. It rolls over at 256. 'HE' in the example is the manufacturer's ID displayed in ASCII characters (if possible). The number after the colon is the last two digits of the UID of the device addressed by this packet .

ERRORS/WARNINGS The list of possible warning messages is listed below.(18.3.1) The two most serious errors or warning are shown in the display.

Time = A time stamp in seconds since the last time the top key was pressed.

```
|REQ [CA]C7h 'HE':91   OK|   | NO ERRORS/WARNINGS |   |CC  PID   PDL  TIME |
|GET   DEVICE INFO    |   |                    |   |20h 0060h 00h 492.09|

|BEFORE  BREAK   MAB |   |SLOT SPACING   AFTER|   |TOT LEN m/us   SLOTS|
|  202u   200u    21u|   |   20-    25    332u|   |   1880u          26|
```

The second display on this line is the entry to a raw packet viewer. Note this display allows viewing slot 0, the Start Code. The following shortcut key are active in this item.
<UP> <DOWN> changes the top line between slot number and slot name
<RIGHT><UP> cycles through display of the bottom line in hex, decimal, or ASCII.
<YES/Q><RIGHT> Jump forward 10 slots
<YES/Q><LEFT> Jump backward 10 slots
<LEFT><RIGHT> Jump to slot 1.

```
| NSC INTERLV     ASC|   |VIEW PACKET RAW?    |   |PARMA DATA          |
|  50               0|   |                    |   |     (EMPTY)        |
```

If the packet being browsed was an RDM response the display will be slightly different. Changes to the first and ninth displays are the ones to note.

```
|RESP[CA]C7h ACK      OK|   | NO ERRORS/WARNINGS |   |CC  PID   PDL  TIME |
|G-R   DEVICE INFO    |   |                    |   |21h 0060h 13h 492.10|

|01234567890123456789|   |01234567890123456789|   |01234567890123456789|
|BEFORE  BREAK   MAB |   |SLOT SPACING   AFTER|   |TOT LEN m/us   SLOTS|
|  332u   177u    29u|   |    0-     9   2153u|   |   2201u          45|

| NSC INTERLV     ASC|   |VIEW PACKET RAW?    |   | PARMA DAT(5 of  19)|
|   0               0|   |                    |   |01h 00h 00h 01h 71h |
```

18.3.1 Warning Messages

The error and warning messages you could see are listed here.

Error	FRAMING ERR SB 2	Framing error - stop bit 2 missing
Error	FRAMING ERR SB 1	Framing error - stop bit 1 missing
Error	PKT TOO SHORT	Packet is shorter than any allowed RDM message
Error	PKT TOO LONG	Packet is longer than any allowed RDM message
Error	BAD SUB-START CODE	The Sub-start code is not 1
Error	LEN/rxCNT MISMATCH	Packet length does not match length given in the packet
Error	LEN/PDL MISMATCH	PDL and the Length field are conflict
Error	CHECKSUM ERROR	Checksum error

Error	WRONG PDL FOR PID	PD length is inappropriate for the PID
Error	BAD PDL FOR RESPTY	PLD is inappropriate for the response type given in the Packet
Error	BAD RESPTYPE	Unknown ACK/NACK type
Warn	INVALID PORT ID(00h)	A controller packet has a port ID of zero
Warn	TN MISMATCH	The transaction number on the response does not match the one the request
Warn	'FRAMING ERR'	Any framing error during discovery
Warn	NON FEh PREAMBLE	Discovery messages have a permeable byte that is not FEh
Warn	DATA OUTSIDE WINDOW	Response data seen after maximum response timeout
Warn	RESP. AFTER NSC/ASC	A response after a null SC or alt SC packet
Warn	RESP. WITHOUT REQ.	A response without Request
Warn	BAD SUBDEV,ROOT ONLY	A root only PID was sent to a sub-device

18.4 Timing Summary

This routine will allow you to review the same timing data you could have viewed live above. However, there is one major difference. Only the packets that meet the match criteria set below will have their timing displayed. By default all packets meet the match criteria. If you are interested in the timings for a packet that meet very specific requirements, setting the match criteria will let you see just those packet's timings. All timings are still available for viewing in the Browse Packet menu and also in the data dumped to the USB port.

18.5 Match Criteria Setup

From a user interface point of view this routine is nearly identical to 16.2 Capture Setup in the advanced RDM menu. However, what they do is quite different. The DMXter4 RDM controller only stores 4 complete packets. It stores a lot information about many packets but not the complete hex dump. For details see section 16.2. Which packets are stored is controlled by the capture setup.

By contrast the sniffer captures up to 500 packets-whatever fits in a 32k RAM. Any packet that meets the match criteria is marked with an the symbol ^{CA} in the browse menu, These packets are marked with an * in USB dumps.

On entry you will see this display.

```
|MATCH PACKETS WITH |
| ANY STARTCODE ▸RDM◂|
```

After you have made this choice you can set up 8 identical slot criteria displays.

```
|  SLOT CRITERION 1  |
|RESP SLOT   0 - ANY |
```

This display allows the selection of a value for any particular slot in an RDM packet. All 8 criteria will be combined in a logical *and* fashion.

* * *

OK, to save a few trees go read the rest of description in section 16.2.

***

After you have set as many of the 8 criteria as you need, you have one other question to answer.

```
|LEN CRIT (INC SC&CS)|   |LEN CRIT (INC SC&CS)|
```

```
|         LENGTH - ANY|   |         LENGTH =  26|
```

Pressing **<UP>** or **<DOWN>** when the cursor is on the dash changes it. Your choices are: = ≠, <, or >. You may then set the numeric argument.

18.6 Sniffer options

We hope these are obvious.

```
|   HIST DATA SIZE   |
|▸50 BYTE◂ UNLIMITED |

|SCOPE TRIG OUT ▸OFF◂|
| REQ RSP DUB CS CAP |

|  DISC TIMEOUT (us) |
|                5600|

|IGNORE PORTID WRNING|
|     ▸YES◂ NO       |
```

Controllers identify which of multiple possible ports that generated this data stream. The number may be set from 1 to 255. 0 is not allowed. However, it is a common error to have this value set to 0. During debugging it may be desirable to suppress this error.

18.7 Send Info to USB

This menu controls which data is sent to the USB port for display on a terminal emulator. The data selections are: dump Packet history, (the basic packet data, dump history with PDATS, dump history with timing, dump history with error details, and history with full HEX dump.

The follow are examples of each type of dump for the same data. We apologize for the small print.

18.7.1 Dump Basic Packet History

```
PACKET HISTORY
 TYPE   TIME TN  DEST/RESPTYPE  DESCRIPTION          CC  PID   PDL  NSC/ ASC
*REQ  361.36 5Bh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   30/   0  OK
*RESP 361.36 5Bh ACK            G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK
*REQ  362.37 5Ch 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   30/   0  OK
*RESP 362.37 5Ch ACK            G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK
*REQ  363.41 5Dh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   31/   0  OK
*RESP 363.41 5Dh ACK            G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK
*REQ  364.45 5Eh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   31/   0  OK
```

18.7.2Packet History with Pdata

```
PACKET HISTORY
 TYPE   TIME TN  DEST/RESPTYPE  DESCRIPTION          CC  PID   PDL  NSC/ ASC      PARAM DATA
*REQ  361.36 5Bh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   30/   0  OK  01h
*RESP 361.36 5Bh ACK            G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK  01h 04h B1h
00h 01h 05h 13h 00h 00h
*REQ  362.37 5Ch 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   30/   0  OK  01h
*RESP 362.37 5Ch ACK            G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK  01h 04h B1h
00h 01h 05h 13h 00h 00h
*REQ  363.41 5Dh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   31/   0  OK  01h
*RESP 363.41 5Dh ACK            G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK  01h 04h B1h
00h 01h 05h 13h 00h 00h
*REQ  364.45 5Eh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   31/   0  OK  01h
```

18.7.3 Packet History with Timing

```
PACKET HISTORY
 TYPE   TIME TN  DEST/RESPTYPE  DESCRIPTION          CC  PID   PDL  NSC/ ASC      BEFORE  BREAK
  MAB  SLOT-SPACE  AFTER TOTLEN SLOTS  PREAM FIRST  LAST TO-NXT
*REQ  361.36 5Bh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   30/   0  OK    202u   200u
  21u    21-   25   366u  1946u    27
*RESP 361.36 5Bh ACK           G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK    366u   193u
  29u     0-    1  2153u  1770u    35
*REQ  362.37 5Ch 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   30/   0  OK    202u   200u
  21u    21-   25   360u  1946u    27
```

```
*RESP 362.37 5Ch ACK            G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK    360u   193u
  29u     0-    1  2153u  1770u    35
*REQ  363.41 5Dh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   31/   0  OK    202u   200u
  21u    21-   25   365u  1946u    27
*RESP 363.41 5Dh ACK            G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK    365u   193u
  29u     0-    1  2153u  1770u    35
*REQ  364.45 5Eh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   31/   0  OK    202u   200u
  21u    21-   25   363u  1946u    27
```

18.7.4 Packet History with Error details.

(No example shown)

18.7.4 Packet History with Hex Dump

```
PACKET HISTORY
 TYPE   TIME TN  DEST/RESPTYPE  DESCRIPTION          CC  PID   PDL  NSC/ ASC      SC SS LEN ------DEST------ -------SRC-------
TN PR MC SUBDV CC PID- PDL DATA----
*REQ  361.36 5Bh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   30/   0  OK  CC 01 19 48 45 00 00 01 92 47 44 00 40 41 69
5B 01 00 00 00 20 02 01 01 01 03 FC
*RESP 361.36 5Bh ACK           G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK  CC 01 21 47 44 00 40 41 69 48 45 00 00 01 92
5B 00 00 00 00 21 02 01 09 01 04 B1 00 01 05 13 00 00 04 DA
*REQ  362.37 5Ch 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   30/   0  OK  CC 01 19 48 45 00 00 01 92 47 44 00 40 41 69
5C 01 00 00 00 20 02 01 01 01 03 FD
*RESP 362.37 5Ch ACK           G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK  CC 01 21 47 44 00 40 41 69 48 45 00 00 01 92
5C 00 00 00 00 21 02 01 09 01 04 B1 00 01 05 13 00 00 04 DB
*REQ  363.41 5Dh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   31/   0  OK  CC 01 19 48 45 00 00 01 92 47 44 00 40 41 69
5D 01 00 00 00 20 02 01 01 01 03 FE
*RESP 363.41 5Dh ACK           G-R  SENSOR VALUE    21h 0201h 09h    0/   0  OK  CC 01 21 47 44 00 40 41 69 48 45 00 00 01 92
5D 00 00 00 00 21 02 01 09 01 04 B1 00 01 05 13 00 00 04 DC
*REQ  364.45 5Eh 'HE':00000192  GET  SENSOR VALUE    20h 0201h 01h   31/   0  OK  CC 01 19 48 45 00 00 01 92 47 44 00 40 41 69
5E 01 00 00 00 20 02 01 01 01 03 FF
```

18.7.5 Hex Dump only

```
PACKET HISTORY
SC SS LEN ------DEST------ -------SRC------- TN PR MC SUBDV CC PID- PDL DATA----
CC 01 19 48 45 00 00 01 92 47 44 00 40 41 69 5B 01 00 00 00 20 02 01 01 01 03 FC
CC 01 21 47 44 00 40 41 69 48 45 00 00 01 92 5B 00 00 00 00 21 02 01 09 01 04 B1 00 01 05 13 00 00 04 DA
CC 01 19 48 45 00 00 01 92 47 44 00 40 41 69 5C 01 00 00 00 20 02 01 01 01 03 FD
CC 01 21 47 44 00 40 41 69 48 45 00 00 01 92 5C 00 00 00 00 21 02 01 09 01 04 B1 00 01 05 13 00 00 04 DB
CC 01 19 48 45 00 00 01 92 47 44 00 40 41 69 5D 01 00 00 00 20 02 01 01 01 03 FE
CC 01 21 47 44 00 40 41 69 48 45 00 00 01 92 5D 00 00 00 00 21 02 01 09 01 04 B1 00 01 05 13 00 00 04 DC
CC 01 19 48 45 00 00 01 92 47 44 00 40 41 69 5E 01 00 00 00 20 02 01 01 01 03 FF
```

18.7.6 Timing Summary

```
RDM REQUESTS              MIN    PREV     MAX
  IDLE BEFORE IN us       202u    202u    202u
  BREAK LENGTH IN us      200u    200u    200u
  MAB LENGTH IN us         21u     21u     21u
  INTERSLOT TIME IN us     20              25
  TOTAL LENGTH IN us     1886u   1886u   1950u
  IDLE AFTER IN us        320u    320u    499u
  NUMBER OF SLOTS          26      26      27
  COUNT      30/   30
RDM RESPONSES
  IDLE BEFORE IN us       320u    320u    499u
  BREAK LENGTH IN us      177u    177u    190u
  MAB LENGTH IN us         12u     12u     12u
  INTERSLOT TIME IN us      0               1
  TOTAL LENGTH IN us     1347u   3806u   3806u
  IDLE AFTER IN us       2153u   2153u   2153u
  NUMBER OF SLOTS          26      82      82
  COUNT      30/   30
RDM BROADCAST REQS.
  IDLE BEFORE IN us       202u    202u    202u
  BREAK LENGTH IN us      200u    200u    200u
  MAB LENGTH IN us         21u     21u     21u
  INTERSLOT TIME IN us     20              25
  TOTAL LENGTH IN us     1886u   1950u   2663u
  IDLE AFTER IN us        202u    202u    202u
  NUMBER OF SLOTS          26      27      38
```

```
  COUNT    168/  168
DISCOVERY RESPONSES
  DISC. FIRST ACTIVITY   338    346    676
  DISC. LAST ACTIVITY   1345   1378   1395
  TIME TO NEXT BREAK    5800u  5800u  5800u
  BYTES IN RESPONSE        9     24     24
  COUNT     58/  163
UNKNOWN RDM PACKETS
  IDLE BEFORE IN us    (EMPTY)
  BREAK LENGTH IN us   (EMPTY)
  MAB LENGTH IN us     (EMPTY)
  INTERSLOT TIME IN us (EMPTY)
  TOTAL LENGTH IN us   (EMPTY)
  IDLE AFTER IN us     (EMPTY)
  NUMBER OF SLOTS      (EMPTY)
  COUNT      0/    0
NULL SC PACKETS
  IDLE BEFORE IN us      202u   202u  2153u
  BREAK LENGTH IN us     200u   200u   200u
  MAB LENGTH IN us        20u    21u    24u
  INTERSLOT TIME IN us    20            25
  TOTAL LENGTH IN us    8498u  8498u 33198u
  IDLE AFTER IN us       202u  >4.1s  >4.1s
  NUMBER OF SLOTS        129    129    513
  COUNT   1993/ 1993
ALTERNATE SC PACKETS
  IDLE BEFORE IN us      182u   182u  >4.1s
  BREAK LENGTH IN us     176u   176u   176u
  MAB LENGTH IN us        13u    13u    14u
  INTERSLOT TIME IN us     0             1
  TOTAL LENGTH IN us    1298u  1386u 22771u
  IDLE AFTER IN us       182u   182u  1719m
  NUMBER OF SLOTS         25     27    513
  COUNT   5411/ 5412
```

18.8 Back to Main Menu

18.9 Clear Sniffer History

There are the two menus left - you can figure it out, right?

*** * * ***

20 THE RECEIVE SCOPE TRIGGER

Note: the timings shown are for the DMXter4 version of this code. The Receive Scope Trigger software is designed for detailed trouble shooting of DMX512 systems and for debugging of new designs. It is not generally needed by show electricians. With Scope Trigger it is possible to trigger an oscilloscope from certain important points within the DMX data stream. Proper use of this feature requires a detailed knowledge of DMX512 and the use of an oscilloscope. When executing Scope Trigger function, the DMXter cannot otherwise receive or analyze DMX512.

This feature consists of two parts, a software module and an optional external printed circuit card. (type number STC1A) Neither is of any use without the other.

20.0.1 Receive Scope Trigger Hardware

The STC1A card provides needed additional hardware to implement Scope Trigger.

Its features include:

* The TTL level trigger signal is on a BNC connector. It is switchable to either a hardware trigger circuit or the software 'arming' signal.
* The TTL level DMX512 data signal is on a BNC connector.
* A delay line in the data output allows viewing of the triggering event.
* EIA485 DMX512 repeater with the ability to optionally invert the data. This driver may be disabled to conserve battery life.
* A self contained, low drain, battery power supply with low battery warning LED.

Functionally the card converts the EIA485 DMX512 signal to a TTL signal. This signal is passed to one input of an 'exclusive or gate' where it is buffered or inverted, depending on the state of a control line from the DMXter. The control line from the DMXter is connected to the other input of the EXOR gate. The output of EXOR is connected to the clock input of a S latch. The S input of this latch is held high. The DMXter provides an 'Arm' signal which is connected to the reset line of the S latch. The DMXter sets the control line to the EXOR gate depending on whether the next trigger is to be on a rising or falling edge of the DMX line. The latch is held in reset until just before a triggering event is expected. It is then released; the next transition of the proper polarity on the DMX line will cause the trigger. After the DMXter software knows the trigger event has passed, it resets the S latch. The arm signal from the DMXter is sometimes also a useful Scope Trigger so it is selectable as the trigger output.

The triggering event to trigger out delay of this hardware is about 25ns. When enabled, the data delay line will add about 75ns of delay to the TTL data output. This should allow you to view the leading edge of the triggering event.

As well as the general resources of the microprocessor and its UART, the Scope Trigger uses certain hardware counters and timers available in this processor to produce highly accurate programmable delays.

20.0.2 Receive Scope Trigger Software

The behavior of Receive Scope Trigger is totally controlled by special software. The Scope Trigger user interface has fewer user warnings and error traps than the general DMXter code. This is because of both the nature of the code and the type of user we expect to use this code. Specifically there is no **'NO DIGITAL INPUT OR INPUT NOT DMX512'** message in Scope Trigger. Also if DMX data stops while the Scope Trigger is waiting for some important event to take place the software will patiently wait there until the event happens. Depending where in the code you are this may cause the user interface to freeze. To regain user interface control, restart the DMX data or exit by way of **<TOP>**.

20.1 TRIGGER ON THE START OF THE BREAK

OVERVIEW

This routine allows you to trigger a scope on the start of a DMX512 break. You will only get a stable trigger on DMX512 transmitters that send packets containing a consistent number of Frames. This routine should work on the vast majority of current production transmitters.

The DMXter arms the trigger card during the stop bits of the last slot of the previous packet. The Scope Trigger card produces a rising trigger when it detects the next falling edge. In a properly formatted DMX data stream that edge will be the beginning of the break. The trigger should be taken from the gated output of the card. This routine is equally useful with either analog or digital storage scopes.

INTERFACE

The entry point is | START OF BREAK? |. On entering you will see the following display

```
|   START OF BREAK    |
|UNSTABLE   SLT:      |
```

If no DMX512 is being received, this display will be steady. If you are receiving DMX512 the number of slots in the packet will appear in the DIM field, and if the number of dimmers in each packet is stable the UNSTABLE field

changes to STABLE. Trigger generation starts after the DMXter determines that the number of slots is stable. A stable display of a console sending 504 slots is

```
|   START OF BREAK    |
| STABLE   SLT: 504   |
```

If the transmitter should switch to a different packet size the STABLE field will momentarily change to **UNSTABLE**, the number in the **SLT** field will change and the display will change back to **STABLE**. The **UNSTABLE-STABLE** field is one shot so even a single packet with a different slot count should be observable. If the transmitter is intermingling packets of different lengths, the field will stay showing **UNSTABLE**. The algorithm used for this trigger mode does not work with changing packet lengths. Note: If you totally lose DMX512 the display will not change, it just acts as if the packet was being sent very slowly. But you should be able to figure it out, you do have a scope connected to the line, don't you?

ALGORITHM DETAILS

The software requires that three packets have the same number of frames for the packet length to be considered stable. The arm signal goes high 1.5µs -1.6µs into the first stop bit of the last frame of the packet. The trigger will be generated on the next falling edge. Obviously no break qualification is possible.

20.2 TRIGGER ON THE END OF THE BREAK

OVERVIEW

This routine is designed to trigger a scope at the end of a break that lasted at least as long as a minimum time set by the user. When the DMXter detects a frame with a framing error that it believes to be a break, it times from the leading edge of that frame: if when the amount of time set by the user has passed we are still in break, the trigger card is armed. The next rising edge of the DMX line will produce a rising trigger on the BNC connector. On analog scopes this is useful for observing the Mark After Break. Additionally on DSO's you may use this routine to observe breaks that cannot be reliably viewed with the routine of 19.1 above. This routine should be reasonably well behaved on most transmitters with either type of scope.

INTERFACE

The entry point is |END BREAK/START MAB?|. On entering you will see the following display

```
|END BREAK/START MAB |
|TRIG ARM AT    63 us|
```

Note the cursor under the 6; you may move it using the **<RIGHT>** and **<LEFT>** keys. Whichever number or space the cursor is under may be edited using the **<UP>** and **<DOWN>** keys. If the cursor is under the one's place the **<UP>** key will increment the number by one with a carry to the ten's place if needed. If the cursor is under the ten's place the **<UP>** key will increment the number by ten with a carry to the hundred's place if needed. Pressing the **<DOWN>** key will decrement the proper place; if an underflow occurs the number will be set to 65µs . The default value for the arm delay is 63µs. Any value up to 16383µs may be selected. Once you have selected a value it will be saved as long as battery power is maintained. All trigger modes other than TRIGGER ON THE START OF THE BREAK use the arm delay and share the same value for it.

ALGORITHM DETAILS

To be considered a possible break a frame must be missing both stop bits, and the data slot must be zero. The line must stay low until the time set by the user has passed. Then the break is considered valid and the arm signal is set. The time is measured from the falling edge at the start of the break. The trigger will be generated by the rising edge. The jitter of the trigger signal is low enough that measurements to 1 µs are reasonably reliable. The trigger signal goes high about 0.5µs before the entered time has elapsed.

20.3 TRIGGER ON THE BEGINNING OF THE START CODE

OVERVIEW

This routine is designed to trigger a scope at the beginning of the START Code if the break has lasted at least as long as a minimum set by the user. When the DMXter detects a break, it times from the leading edge of that break; when the amount of time set by the user has passed, the trigger card is armed. The next falling edge of the DMX line will produce a rising trigger on the BNC connector. This is useful for observing the START Code and as a general trigger at the beginning of a packet.

INTERFACE

The entry point is |BEGIN OF START CODE?|. On entering you will see the following display

```
|BEGIN OF START CODE |
|TRIG ARM AT    63 us|
```

The interface behavior is identical to that for **END BREAK/START MAB**.

ALGORITHM DETAILS

Other than generating a trigger on the falling edge this routine is identical to the **Trigger on the End of the Break.**

20.4 SLOT TRIGGER

OVERVIEW

The SLOT TRIGGER routine is actually a number of powerful trigger routines selectable from a bar menu. The main thrust of these routines is to allow you to trigger on any slot in a DMX512 packet. The trigger is generated when the 'AND' of three conditionals is true. An important thing to keep in mind is that the trigger is generated only AFTER a slot in the DMX packet has satisfied all of the conditions.

The qualifiers for the START Code are: equal(=), not equal (≠), or don't care (----).
For slot number they are: equal (=), or don't care, (----).
For slot level they are: equal (=), greater than (>), less than (<), not equal(≠), or don't care (----).

Getting these routines to do what you want will require careful understanding of what they do. Unlike the other trigger routines all of these routines cause the receiver to read the DMX data and store it in the slot table. All of these routines will run in either a continuous mode or a single shot mode. In the continuous mode a trigger is generated every time the condition is met; in the single shot mode only one trigger is generated. In the continuous mode all packets are written to the slot table; in the single shot mode, reception stops at the end of the first packet that satisfies the conditional trigger.

INTERFACE

The entry point is SLOT TRIGGER? |. On entering you will see the following display

```
|MIN BREAK IS    63us|
|     CHANGE IT?     |
```

This allows you to set the shortest break that may be received for a packet to be further analyzed. After you have changed the break or bypassed doing so you will enter the main bar menu.

```
|STCD  SLT  LEV  CAPT|
|---- =  1 ----  CONT|
```

20.4.1 Triggering after a Slot

These are the default settings and this is the most common mode of operation. These settings cause a trigger to be generated on every packet, regardless of START Code, after slot 1. On an analog scope or DSO set to view post trigger you will see the start bit of slot **2**.

The slot number may be set from 0 to 512. To set the slot number, move the underline cursor with the **<RIGHT>** and **<LEFT>** keys. Place the cursor under the digit you wish to change. If the cursor is under the one's place the **<UP>** key will increment the number by one with a carry to the ten's place if needed. If the cursor is under the tens place, the **<UP>** key will increment the number by ten with a carry to the hundred's place if needed. Pressing the

<DOWN> key will decrement the proper place; if an underflow occurs the number will roll over to the highest allowed number, in the case of slot 512.

If you want to view slot 1, set the number to 0. Setting the number equal to the number of slots sent will cause a trigger on the start of the break of the next packet. The reason we display the slot that causes the trigger and not the slot that will be viewed is so that we may have consistency with the rest of the slot trigger modes. If one is looking at the next slot the trigger will almost always be taken from the gated trigger signal from the BNC connector. If you are using a DSO to look backward in time at the slot that caused the trigger you may find that the arm signal gives less jitter. The difference between these two signals is that the Arm signal is precisely delayed from the start bit of the arming slot, while the gated trigger is synchronous with an edge in the next slot.

TIMING DETAILS
With the START Code and the level entries set to 'don't care' the delay from the rising edge of the stop bit of the triggering frame to the generation of the arm signal is 6.4µs.

20.4.2 Trigger on Packets with START Code 'x'

STCD stands for START Code. Setting the cursor under any one of the STCD spaces and pressing **<UP>** will cause the START Code to come out of 'don't care'. On entry the START Code will be set to the DMXter present START Code setting, generally

```
|STCD  SLT  LEV  CAPT|
|=  0 =  1  ---- CONT|
```

Now the trigger will be generated only for packets that have a zero START Code. Any slot number may be selected. See the timing details below.

The allowed qualifiers for a START Code are equal and not equal. The latter may be used with the single shot mode (**ARM**) to capture suspect corruptions of the START Code.

Try placing the **SLT** field into the 'don't care' state. Do this by placing the cursor under the = sign and pressing either **<UP>** or **<DOWN>** keys. You will note that **LEV** field comes out of 'don't care'. Only one of the SLT or LEV fields may be in 'don't care' at the same time.

TIMING DETAILS
The delay from the rising edge of the stop bit of the triggering frame to the generation of the arm signal depends on which slot generates the trigger. If we are triggering on slot 1 through 512, the delay is 6.4µs. For slot 0 the delay is ~ 4.8µs .

20.4.3 Triggering If Any Slot Is at Level 'X'

Leave the **SLT** field in the 'don't care' state, select the START Code value you want, including 'don't care'. The setting of the START Code will determine which packets will be checked for levels. This mode is novel in that multiple triggers may be generated by a single DMX packet. Each slot is evaluated and a trigger is generated whenever the qualified level is matched. The qualifiers for levels are: equal (=), not equal (≠), level greater than (>), and level less than (<).

This is a mode where the fact that a trigger occurred may be all you wish to know, so consider using the single shot mode. If multiple triggers occurred you may be more interested in where they were than what the data was. You might consider viewing the trigger signals directly.

TIMING DETAILS
The delay from the rising edge of the stop bit of the triggering frame to the generation of the arm signal is about 8.4µs. So with fast DMX devices you may need to use the Arm signal.

20.4.4 Triggering Slot'X' Is at Level 'Y'

If you enable both the **SLT** and the **LEV** fields at once the trigger will be generated after the indicated slot if it meets the level restrictions. The delay for this signal is about 7µs so you can use the gated output for most triggering.

20.4.5 Using the One Shot Mode

The single shot mode is controlled by the last field. Placing the cursor under any one of the bottom line spaces beneath **CAPT** and pressing either **<UP>** or **<DOWN>** will change the **CONT** flag to **ARM**. This flag will stay showing **ARM** until the trigger conditions are met, then it changes to **TRIP**. At that time a single trigger is generated and the packet containing the trigger is preserved in the slot table. At this point you may wish to temporarily leave the **SLOT TRIGGER** to view the captured levels. You may do this by pressing **<YES/Q> <DOWN> <YES/Q>**. You may return to the **SLOT TRIGGER** without losing your setup with the one exception that the **TRIP** flag will be replaced by the **CONT** flag.

20.4.6 USING HEX NUMBERS IN RECEIVE SCOPE TRIGGER

If the DMXter is set to display in hexadecimal, the START Code and slot levels will be displayed as a two-digit hex number followed by a lowercase 'h'.

20.5 VIEW CAPTURED LEVELS

OVERVIEW

This routine allows you to view data stored in memory by the SLOT TRIGGER software above. The data that will be displayed is the last packet received. If you have not run SLOT TRIGGER since you entered the Receive Scope Trigger menu, the data in the slot table will be whatever was left from the last time transmit or receive functions of the DMXter were used. The only Scope Trigger routine that writes slot level to the slot table is SLOT TRIGGER.

INTERFACE

The entry point is |VIEW CAPTURED LEVEL?|. The interface for this routine is the same as VIEW LEVELS.

20.6 FRAMING ERROR TRIGGER

OVERVIEW

The **FRAMING ERROR TRIGGER** has a dual nature. If either or both of the two stop bits are missing from a frame and the data slot is not zero, a trigger is generated. No further time qualification is required. If both of the stop bits are missing, the data slot is zero, and the line goes high (marking) before the time set by the user, a trigger is generated. The trigger pulse is generated when the time delay expires. In many ways this is the inverse of the minimum break qualification routines (above) that require that a break lasts at least as long as the time set by the user for a trigger to be generated.

INTERFACE

The entry point is |FRAME ERROR TRIGGER? |. On entering you will see the following display

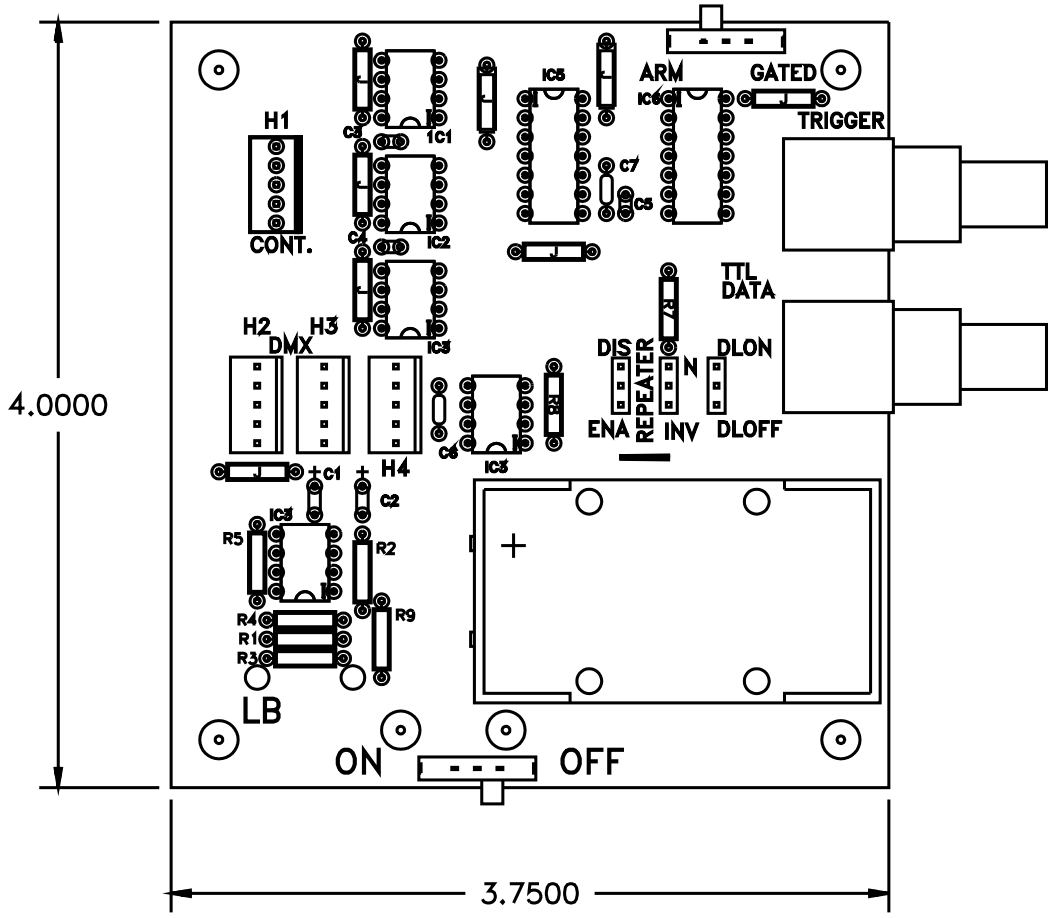
```
|FRAME ERROR TRIGGER |
| ERROR <     63 uS|
```

This routine has a number of uses in tracking down glitches on a DMX512 line. Another use is to generate a trigger signal a precise time after the start of a DMX packet.

For normal use the gated trigger output produces a pulse with the needed accuracy. Using the arm signal may produce slightly more predictable timing. The framing error trigger is output about 4µs after the timer runs out. The trigger output is a short positive going pulse lasting approximately 2.5µs.

20.7 FURTHER HARDWARE DETAILS

Header Connections:



H1, Control: This header carries the Scope Trigger control signals from the DMXter to the STC1A card. It should be wired to an A5M connector plugged into the **DMX-512 OUT** connector on the Scope Trigger equipped DMXter. All header to 5 pin cables should be wired one for one.

1	Shield
2	- Arm
3	+ Arm
4	- Phase
5	+ Phase

H2 - H3, DMX512 Data: Headers H2 and H3 are wired in parallel. H2 should be wired to the DMX source under test. H3 should be wired to an A5F connector plugged into the **DMX-512 IN** connector on the Scope Trigger equipped DMXter.

1	Shield
2	- DMX
3	+DMX
4	- Aux.
5	+ Aux.

H4 Repeater: This connector is the output of the DMX repeater. In Scope Trigger mode the DMXter always terminates the DMX512 line. Hence if you need to simultaneously use the DMX signal under test and cannot tolerate double line termination you will need to use the built in repeater. The repeater is controlled by two sets of programming jumpers. The **DIS**able - **ENA**ble jumper block controls whether the repeater is enabled or tri-stated. The **N**ormal - **IN**verted jumper block controls whether the data is passed normally or inverted. This repeater is shipped disabled since its use shortens battery life. The scope card draws only about three to 4 MA. Driving a terminated DMX line draws about 25 MA additional.

1	Shield
2	Repeater out -
3	Repeater out +
4	Aux. - (jumped to H3-4)
5	Aux. + (jumped to H3-5)

The Delay line: The jumper block marked **DLON - DLOFF** controls whether a deliberate delay of about 75nS is introduced into the DMX data output on the BNC connector. Units are shipped with this delay enabled.

Getting The ARM Signal:
On the current version of the Scope Trigger card (STC1A R2)a switch on the edge of the card near the trigger BNC connector selects whether the **GATED** trigger or the **ARM** signal is available on the BNC connector.
*** * * ***

21 USB Dongle Mode

The Dongle mode provides an external application with access to RDM request generation and response reception via USB. It allows the external application to read the DMXter generated results code. See Table 16.6. Dongle mode is entered by going to either main menu item, RDM CONTROLLER or RDM INSERTER, and holding down the **<LEFT>** and **<RIGHT>** keys while pressing and releasing the **<YES/Q>** key

21.1 **RDM Integrity** Test software

The dongle mode interface is optimized for use with a software package written by Benjamin Electric and sold by Goddard Design Co. The Benjamin Electric **RDM Integrity** test software application can test an RDM responder's conformance to the syntax of RDM by sending a sequence of requests and checking the responder's reply. The list of tests to be run may be selected to test a specific PID or groups of PIDs that control a functional area. When **RDM Integrity** is run with any DMXter4 product, packet timing limits may be checked on all responses. The packet timing uses the industry standard timing routine implemented by the DMXter. For more details see the **RDM Integrity** data sheet.

RDM Integrity can be run on most recent versions of Windows (XP -7- 8 and we believe on Vista) To run the package you will need a supported USB to RDM interface and a PC. The following interfaces are supported:

MiniDMXter4	Goddard Design Co.
DMXter4 RDM	Goddard Design Co.
DMXter4A RDM.	Goddard Design Co.

JESE RDM-TXI MK2 James Embedded Systems Engineering
JESE RDM-TRI MK2 James Embedded Systems Engineering

The DMXter device must be running v4.30(DMXter) or V3.30(MINI) or later code. Upon payment of the licencing fee your DMXter will be registered for use with the program. The application may be run on any PC without registration of that PC. It is the DMXter that is the 'seat'.Note that available feature sets may change depending on which DMXter model is being used.

21.2 Open Lighting Architecture test software

.The Dongle mode can also provide a 485 layer interface for the Open Lighting Architecture test software. Our V4.18 software operates with OLA Version 0.8.11 and may work with later versions of their RDM test software. Users having questions regarding use of OLA software need to contact OLA. Goddard Design does not provide support for the OLA software. We do not guarantee hardware compatibility with that software.

*** * * ***

22 COLORTRAN PROTOCOL OPTION

You may order a DMXter4 with an option that allows it to send and receive Colortran's proprietary digital protocol. This protocol is usually referred to as CMX. It is the parent protocol on which DMX512 was based. The primary difference between CMX and DMX512 is that CMX uses a baud rate of 153.6K while DMX512 uses a baud rate of 250K. A side note: the baud rate of CMX has often erroneously been listed as 156K.

This option should be of great use to anyone servicing systems that use this protocol. Many of the DMXter4's features support CMX, but certain differences must be taken into account.

Colortran is field retofittable on DMXter4. This option is also retrofittable to all existing *Lil'*DMXters, but it requires that the unit be returned to the factory for additional hardware.

22.1 HOW TO IDENTIFY CMX EQUIPPED DMXTER4

A DMXter4 fitted with this option is identified by a 'C' before the software version number. eg. C4.15.

22.2 NAMING CONVENTIONS FOR THE CMX PROTOCOL

The DMXter4's software uses either COLORTRAN or 'CTN' in its display messages to identify the CMX protocol. The reason for this is that at a quick glance CMX and DMX are easily confused in the block letter character set of the LCD display. This naming change is done only for clarity.

22.3 SELECTING THE CMX PROTOCOL

The primary standard of units fitted with this option is still DMX512. Units so fitted must be switched via software to mode. Once switched they will stay that way until switched back or until the power-up defaults are restored.

There are two methods of changing the unit to mode. One is by way of a switch in the SETUP OPTIONS menu. This is a bidirectional switch which will offer the user whichever standard the unit is not currently set for. If the unit is set for DMX512 the display will read:

```
|    DATA IS DMX     |
| SET FOR COLORTRAN  |
```

Using this switch will set the START Code to zero. Returning the switch to DMX also resets the START Code. The other method is a new 'flavor' in the **TRANSMIT DMX512**, **SEND FLAVOR** submenu.

```
|   SEND FLAVOR?     |
|    CMX 153.6k      |
```

The following should be noted: While DMX512 flavors only affect transmitted DMX, the **CMX 153.6k** flavor sets the DMXter4 to transmit and receive CMX. Also there is only one transmit flavor available for CMX. The values for this flavor are listed below. Using this switch will set the START Code to zero.

22.4 HOW TO TELL IF A DMXter4 IS SET TO CMX PROTOCOL

If you have pressed **<TOP>** the DMXter4 is sitting on the Transmit menu; the display will be changed if the unit is set to CMX.

```
|     MAIN MENU      |
|TRANSMIT COLORTRAN? |
```

The Receive menu item also changes to:

```
|     MAIN MENU      |
| RECEIVE COLORTRAN? |
```

The displays for other **MAIN MENU** items do not change when the protocol is switched. But all of these functions will now support protocol.

22.5 CHANGES TO TRANSMIT MENU ITEMS

Any Transmit menu item that has a first line that normally reads TRANSMIT DMX512 will change to read TRANSMIT Colortran.

The SEND/EDIT SNAPSHOT routine display matrix is changed. The first example below is a possible display of a DMXter4 without the CMX option.

```
SLT:   1   2   3   4
LEV:  98  FF  50   0
```

The following examples are for units fitted with the CMX option. When the protocol is set to DMX512 the display will be as shown below. The field that used to read LEV is changed to read DMX to indicate the current protocol setting.

```
SLT:   1   2   3   4
DMX   98  FF  50   0
```

When the protocol is set to CMX the display will be as shown below. The field that used to read LEV is changed to read CTN to indicate the current protocol setting.

```
SLT:   1   2   3   4
CTN:  98  FF  50   0
```

22.5.1 The Change Send Flavor Submenu & CMX

```
| TRANSMIT COLORTRAN  |        | TRANSMIT DMX512   |
| CHANGE SEND FLAVOR?|        |CHANGE SEND FLAVOR?|
```

On a DMXter4 equipped with the option there is an additional flavor entry. It is the last selection. Hence, it is

```
|    SEND FLAVOR?    |
|     CMX 153.6k     |
```

Selecting this flavor sets the unit, **including resetting the START Code, to zero.** Using the **<UP>** or **<DOWN>** keys to move to another flavor, accept that flavor by pressing the **<YES/Q>.** Selecting **a DMX flavor does not reset the START Code** to zero. However, there is no reason that it should be other than zero.

22.5.2 Changing the START Code While in CMX Mode

The submenu item that allows the DMXter4 to set the START Code to non null values is available when the unit is in CMX mode. It is left active to keep the unit's behavior as similar as possible in both protocol modes. We know of no valid CMX uses where the slot used as the START Code in DMX is anything but a null. Therefore we doubt that you will ever need this feature in CMX.

Note that whenever the protocol is changed either from DMX to CMX or CMX to DMX, the START Code is reset to a null (zero) value.

22.6 CHANGES TO RECEIVE MENU ITEMS

Any Transmit menu item that has a first line that normally reads RECEIVE DMX512 will change to read RECEIVE COLORTRAN.

The VIEW LEVELS routine display matrix has been changed in the same way as the SEND/EDIT SNAPSHOT display. The LEV characters have been replaced by CTN.

22.7 CMX View Parameters Works the Same as in DMX

If you have used the CMX feature on a ***Lil*'DMXter** you may remember that you had to use a correction factor with some of the measured parameters. This not the case with DMXter4. The measurement routines are identical for either protocol.

22.8 COLORTRAN CMX TIMINGS, AND GDC'S CMX FLAVOR

The following section gives in tabular form some of the important timing information for CMX.

	Ideal Times	DMXter4 times
CMX Baud Rate	153.6 k Baud	150 k baud
CMX Bit Time	6.5104 µs	6.667 µs
CMX Frame Time	71.615 µs	73.33 µs
BREAK		214.8µs
MAB		19.53µs
Break to Break		40316µs
Slots Per Packet	512	512
Baud Rate Error		-1.9%

The DMXter4 can produce a baud rate that is close to ideal for CMX protocol. It cannot produce the exact baud rate. The baud rate is within workable tolerances.

22.9 CMX FLICKER FINDER

The CMX version has the same display and is operationally identical to the DMX version. The test is run at the CMX baud rate.

22.10 CMX CABLE TESTER

The CMX version is operationally and display identical to the DMX version. The test is run at the CMX baud rate. This means that some cables may pass the CMX data test that would fail the DMX data test. This is appropriate since CMX makes lower demands of its cable.

22.11 CMX SHOWSAVER

The operation of CMX ShowSaver is identical to the DMX version. The only display difference is that when editing levels the LEV characters are changed to CTN as they are in SEND EDIT.

Since changing protocols does not change any recorded ShowSaver looks it is possible to record looks from a console set to one protocol, say DMX512, and then switch protocol to the other to play them back. This could get you out of a very tight spot someday.

If the DMXter4 is set to enter DMX (CMX) Monitor mode and receives data sent on the protocol that it is not set for, it will act just as if it saw no data at all. No additional indication of a problem is given.

22.12 ROUTINES INCOMPATIBLE with COLORTRAN

The following are incompatible with Colortran;

DMX512 TEST PACKET - HEX 55	Sec 4.12.4
DMX text packet - Ascii or UTF-8	Sec 4.12.5 &
SIP	Sec 4.12.7
All RDM functions	All Sec 15, 16, 17