OPERATING MANUAL- SECTI0N 2
FOR

DMXter4™ RDM

Software V 4.10 - DRAFT
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

 9/10/10
L:\word_p\wp6doc\dmx-man\dmx4.10_sec2.wpd

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 packet 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) C_A ='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.

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)

4) Time = A time stamp in seconds since the last time the top key was pressed.
This display is for the response to a DMXter request. We guess that it the Device Under Test that you are most interested in.

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
```

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 on the oldest packet in the browse memory will have a display with most of its flag fields filled in.
This display highlight information about the packet.
(R) An R in the bottom row means that this packet is a retry.
(Q) A Q in the bottom row means that this packet is the response to a 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 out going 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?

16.1.7 View Raw Captured Response Packet

Do we need to say more?

16.1.8 View Raw Previous Request Packet

The last packet sent is stored and may be inspected by this menu item.

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.

16.2 Capture Setup

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

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 or 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 criterion 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 is 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. The last item lets you reset all of the capture criterion to their clear state.

16.3 Build a Custom Request Packet

The following x 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. . .

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

The final two items lets you send the packet you just built and see what comes back.

16.3.3 Viewing the response.

This lets you view the complete returned packet.

16.3.4 Send Packet Repeatedly

The DMXter allows sending a packet repeatedly. The Packet to be sent is the last packet edited in the custom packet software. If you simply go directly to this menu item the packet will be whatever RDM packet was sent last. After entering the menu item you will see the top line below. The packet count will be rising quickly. The second line will change as you scroll <DOWN>.

Top line	`|PKT:  15 SENDING ↕|`	
On entry	`|COR:  TOUT:  |`	Corrupt packets - Timedout packets
Down 1	`|NAK:  ACKT:  |`	Naked packets - ACK timer
Down 2	`|ACK:  ACKO:  |`	Acked Packets - Ack Overflow
Down 3	`|  Captured:  |`	Number of packets that caused a capture.
Down 4	`|PDL Range ### to ###|`	Shows the min and max PDL size

For Discovery packets only the next items will be displayed instead.

`|ACT:  IDLE:  |`	For DUB packets only
`|COL:  CSOK:  |`	
`|---:  FERR:  |`	

16.4 Check for Broadcast Response

A broadcast packet is sent to all responders. A vendorcast packet is sent to all responders built by one vendor. Neither packet type should be responded to. However it is a bug that can creep into a design. When you press **<YES/Q>** on this item all you hope to see is 'SENDING' and after a short pause 'GOOD'.

What the DMXter has does is, it first sends the listed packets to the Broadcast address. It then sends the same 4 packets to the Vendorcast address using the vendor code of the currently selected device.

Discovery Mute
Get Device Info (Gets are not sent to the broadcast address but if they are they should be ignored)
Set Identify Off
Get PID 6123h (This is an unknown PID which checks that error handler responds correctly broadcast.

A response to any one of these packets prints a warning, similar to below.

```
|12345678901234567890|
|0060:DEVICE INFO    |
|RESPONSE TO BROADCST|
```

16.5 Setup Scope Trigger Output

The DMXter4 can generate a pulse on RDM packets that meets 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.

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

▸REQ◂ A trigger is generated about 1.3µs after the start of the break for the request packet.
▸REQ RESP◂ A trigger is generated about 1.3µs after the start of the break for the request packet and ends within a microsecond of the DMXter driving the line to Mark before Break of the next packet after the response.
▸RESP◂ A trigger is generated about 500µs before the line is tri-stated waiting for the response.
▸CAPT◂A trigger is generated for any packet that meets the capture requirements discussed above. It is generated within a few hundred microseconds after the response is complete. Note: when the response is considered complete depends on the setting of '15.10.6 End Response Reception'. It will trigger sooner in most cases if reception is terminated on byte count. If the packet is terminated by time that time is controlled by '17.1 Lost Packet Timeout'

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 reasoner 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	00		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						
	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			

16.6 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

```
|TIMING FLAVOR:▸SLOW◂|
|  RDM   USER: A B C |
```

RDM Controller Flavors						
	Break	MAB	Inteslot Time	MBB	Slots per Packet*	
Slow	200µs	22µs	20µs	198µs	512	
RDM	176µs	14µs	0µs	178µs	512	
User A#	160µs	20µs	20µs	33ms	128	user setable
User B#	160µs	22µs	20µs	880 µs	512	user setable
User C#	160µs	22µs	20µs	16ms	512	user setable

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

Section	Default	
17.1 Lost Packet Timeout	3440µs	Table 3.2 line 5 + the maximum Break and MAB[1]
17.2 Slot Timeout	2144µs	Table 3.3 line 1, plus 1 slot time. This added slot time is required by how we measure this timing.
17.3 Discovery Timeout	5800µs.[2]	Table 3.2 line 2
17.4 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 LINE VIEW an RDM SNIFFER

This section of the manual is under construction

19 THE RECEIVE SCOPE TRIGGER

Note: the timings shown are for the DMXter2 version of this code. Timings for the DMXter4 RDM version will be added soon.
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.

19.0.1 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.

19.0.2 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>**.

19.1 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 2.5µs - 2.9µs 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.

19.2 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    65 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 65µs. 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. To allow for worst case latency in the break time the timer is offset by a small amount. This latency has a certain amount of jitter. On most packets if the trigger is set to arm at 65µs the arm signal will actually go high 63.5µs after the leading edge of the break. Generally this will mean that the break is one micro second shorter than the maximum setting that gives a break trigger. If the break starts to be displayed as the delay is decreased from 90 to 89 µs the true break length is 88µs.

19.3 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    65 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.**

19.4 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    65uS
  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
```

19.4.1 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 3µs.

19.4.2 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, but there are timing limitations on triggering on slot zero in this mode, meaning that for general viewing it is better to start with slot 1. 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 3µs.

If the trigger is set to SLT the delay is 11.6µs. On an analog scope you may lose part of the first slot after the trigger depending on the amount of interslot time in the packet. You will lose less if you switch to triggering on the arm signal.

19.4.3 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. Certain timing restrictions also must be pointed out. (Yeah, we wish we didn't have them too.)

TIMING DETAILS

The delay from rising edge of the stop bit of the triggering frame to the generation of the arm signal with the level check set for = or ≠ is 16.8µs. If the level check is set for < or > the delay is 19.6µs.

19.4.4 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.

19.4.5 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.

19.4.6 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'.

19.5 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.

19.6 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

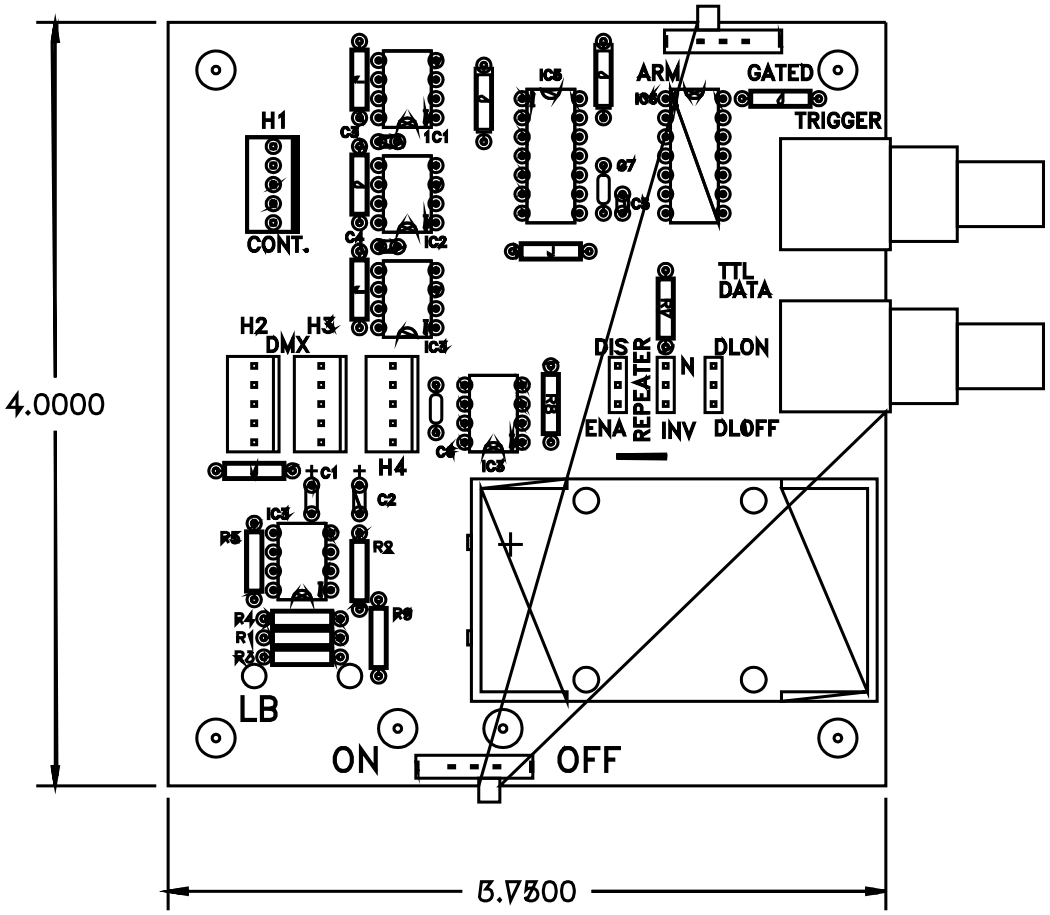
```
FRAME ERROR TRIGGER
  ERROR <       65 uS
```

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.

19.7 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.

20 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.

20.1 HOW TO IDENTIFY CMX EQUIPPED DMXTER4S

A DMXter4 fitted with this option is identified by a 'C' after the software version number.

20.2 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.

20.3 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.

20.3 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.

20.4 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
```

20.4.1 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.

20.4.2 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.

20.5 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.

20.6 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.

20.7 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.

20.8 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.

20.9 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.

20.10 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.

20.11 ROUTINES INCOMPATIBLE with COLORTRAN

The following are incompatible with Colortran;

ShowPlayer
DMX text packet
SIPs
RDM

```
** TEXT PACKETS
** The purpose of the ASC text packet is to allow equipment to
** send diagnostic information formatted for display.
** The START Code is    17h
** Packet length 3 thru 512
**(However for timing reasons most packets will should be padded to a
** minimum of 24 data slots.)
** Slot allocation is as follows:
**          slot 0: START Code 17h
**          Slot 1: Page number of one of the possible 256 text pages.
**          Slot 2: Characters per Line.
**                This Indicates the number of characters per line that the
**                transmitting device has used for the purposes of formatting
**                the text. A slot value of zero indicates ignore this field.
**    Slots 3-512: ASCII text
**                All characters are allowed and where a DMX512 text viewer
**                is capable, it shall display the data using the ISO/IEC 646
**                standard character set.
**    A slot value of zero (ASCII Null) shall terminate the ASCII string.
**    Slots transmitted after this null terminator up to the reset sequence
**    shall be ignored.
*****************************************************************************
```

Text packets sent by the DMXter4 are fixed. We allow for 8 messages.
Details of the DMXter4 format are:

Slot 1 is set to 00h thru 08h to identify the current message.
Slot 2 is always sent as 00h.
Slot 3-511 are sent as ASCII text as required. After the last ASCII character is sent, a Null will be sent. If fewer than 25 slots are sent, the packet will be padded out to 25 slots by whatever garbage is in the transmit buffer. These characters should be ignored.
Slot 512, if sent, will always be a Null.
In the listings below the text to be sent is delimited with single quotes (').

```
;text packet strings
;                |         1         2         3         4
                 |12345678901234567890123456789012345678901
TEXT_MS0         ;
            DB   'DMXter4 RDM v4.00'
TEXT_MS1
            DB   'COPYRIGHT GDC 2009'
TEXT_MS2
            DB   'ESTA DMX512A TXT PACKET'
TEXT_MS3
*                |         1         2         3         4
*                |12345678901234567890123456789012345678901
            DB   'Alice was beginning to get very tired of'
            DB   ' sitting by her sister on the bank, and '
            DB   'of having nothing to do: once or twice s'
            DB   'he had peeped into the book her sister w'
            DB   'as reading, but it had no pictures or co'
            DB   'nversations in it, "and what is the use '
            DB   'of a book," thought Alice, "without pict'
            DB   'ures or conversations?"  So she was cons'
            DB   'idering, in her own mind (as well as she'
            DB   ' could, for the hot day made her feel ve'
            DB   'ry sleepy and stupid), whether the pleas'
            DB   'ure of making a daisy-chain would be wor'
            DB   'th the trouble of getting up '
```

Message 4 below is most of the common punctuation listed in numerical order.

Some displays may not provide readable results with all of these characters.

```
TEXT_MS4
      DB    021   ;     !     (exclamation mark)
      DB    022   ;     "     (double quote)
      DB    023   ;     #     (number sign)
      DB    024   ;     $     (dollar sign)
      DB    025   ;     %     (percent)
      DB    026   ;     &     (ampersand)
      DB    027   ;     `     (single quote)
      DB    028   ;     (     (left/opening parenthesis)
      DB    029   ;      )    (right/closing parenthesis)
      DB    02A   ;     *     (asterisk)
      DB    02B   ;     +     (plus)
      DB    02C   ;     ,     (comma)
      DB    02D   ;     -     (minus or dash)
      DB    02E   ;     .     (dot)
      DB    02F   ;     /     (forward sl

      DB    03A   ;     :     (colon)
      DB    03B   ;     ;     (semi-colon)
      DB    03C   ;     <     (less than)
      DB    03D   ;     =     (equal sign)
      DB    03E   ;     >     (greater than)
      DB    03F   ;     ?     (question mark)
      DB    040   ;     @     (AT symbol)

      DB    05B   ;     [     (left/opening bracket)
      DB    05C   ;     \     (back slash)
      DB    05D   ;     ]     (right/closing bracket)
      DB    05E   ;     ^     (caret/cirumflex)
      DB    05F   ;     _     (underscore)
      DB    060   ;     `

      DB    07B   ;     {    (left/opening brace)
      DB    07C   ;     |    (vertical bar)
      DB    07D   ;     }    (right/closing brace)
      DB    07E   ;     ~    (tilde)
```

Message 5 is some common ASCII formatting characters. Many LCD displays do not interpret these characters as a dumb terminal would. Some displays, including the DMXter4 allow custom characters to be mapped to these and other ‘unused’ codes. Care should be taken when choosing non alphanumeric characters.

```
TEXT_MS5:
            DB    '<BS>'
            DB    07h
            DB    '<TAB>'
            DB    09h
            DB    '<LF>'
            DB    0Ah
            DB    '<CR>'
            DB    0Dh
```

Message 6 sends a single character ‘S’ that is meant for testing the behavior of receivers when sent a very short string.

```
TEXT_MS6:   DC    'S'
```

Message 7 tests the behavior when a short string is terminated by a null but additional characters are sent after the null. A properly set up display should see only the character 'T'. If you cursor over the null character, the DMXter4 will display the rest of the string; but if given the whole string, it will stop receiving at the null.

```
TEXT_MS7:   DC    'T',00h,'You should not see this message,'
            DC    'it''s after a null'
```