DecaBox - RS232 to DMX and MIDI Simultaneously

Record & play back DMX data, transmit MIDI, and more, all with a simple RS-232 syntax.

Basic System Information

This firmware personality for the DecaBox receives RS-232 data and allows full control of a DMX 512 universe, including scene snapshots, dynamic captures and more. Further, via RS-232, MIDI data may be transmitted to downstream devices.

- The system ships with an international switching power supply. System power requirements are 9-12v DC, center positive, 2.0 amps. The power supply connector has standard dimensions of 2.1mm x 5.5mm.
- MIDI data is passed transmitted via the MIDI OUT 5-pin DIN connector.
- DMX lighting data is generated on the Neutrik 5 pin XLR female jack. If necessary, a 5 pin to 3 pin adapter cable may used to connect various lighting fixtures.

> The DecaBox USB port is used for firmware updates. *It does not accept MIDI data.*

> Originally, the lighting guys wanted to keep their wiring separate from the audio crew, who were using XLR-3 microphone cable; thus the 5 pin lighting data standard. However, in nearly every current implementation of DMX control only pins 1, 2 and 3 are used. The 5 pin connectors cost about $2 more in quantity, so some manufacturers eschew them for less expensive 3 pin versions. Professional and touring gear still relies nearly exclusively on the 5 pin infrastructure. In either case, pin 1 is ground, pin 2 is ‘data complement’ or D- and pin 3 is ‘data true’ or D+.

Overview

> System baud rate is **9600 8N1.** That's 8 data bits, no parity, one stop bit. If testing with a PC, a null modem / crossover cable is **required**. Pins 2 & 3 must swap when two 'master' devices are talking with each other. Pin 5 is ground.

> In the following pages, the text **[cr]** is used to represent a single data byte, the carriage return. This value is decimal 13 or hex 0x0D. All commands are terminated with a carriage return.

Also, in the following pages some syntax may be provided in code blocks, like this:

```
This is code in a block.
It's how our online markdown language calls out this type of text.
--! Note the grey '1' and '2', etc in the far left column. !--
These are line numbers for reference only. Don't include
them in actual commands!
Some sample real RS-232 commands just for reference:
C[cr]
F000@255:000[cr]
```

Screenshots of terminal output are based on the excellent RealTerm program.

Send MIDI Data via RS-232

Overview

Most MIDI messages are sequences of 2-3 8-bit bytes. The largest hurdle when integrating the DecaBox with Crestron / AMX / Control4 equipment is properly formatting these messages. An excellent, detailed description of their syntax can be found here:

- https://www.midi.org/specifications/item/table-1-summary-of-midi-message
- https://www.midi.org/specifications-old/item/table-3-control-change-messages-data-bytes-2

> In our documentation, we use this format to describe raw byte data: `$AA $BB $CC` . The $ sign signifies hexadecimal values, which means that AA, BB, etc are in the range [00 FF] hex, which maps to [0 255] decimal. Spaces are inserted between bytes in this documentation merely for convenience in reading.

MIDI messages are zero based and *generally* bytes 2 and 3 have a maximum value of $7F, or 127 decimal. A simple online decimal <-> hex converter can be found at https://www.binaryhexconverter.com/decimal-to-hex-converter

Here are sample MIDI messages:

```
$90 $01 $7F   Note on, MIDI channel 1, note #2, full velocity $8F $07 $40   Note off, MIDI
channel 16, note #8, 50% velocity$B1 $03 $10   MIDI CC #4 (foot controller), ~30% intensity,
MIDI channel 2
$F0 $01 $02 $03 $04 $05 $06 $F7   A short MIDI sysex message. These are framed by [F0 F7] always.
```

(The MIDI channel is the lower nibble of the first byte. That is, $BX means MIDI CC message and X can vary between [$0 $F]. Hex $0 means MIDI channel 1, $1 means MIDI channel 2, etc. $F is channel 16.)

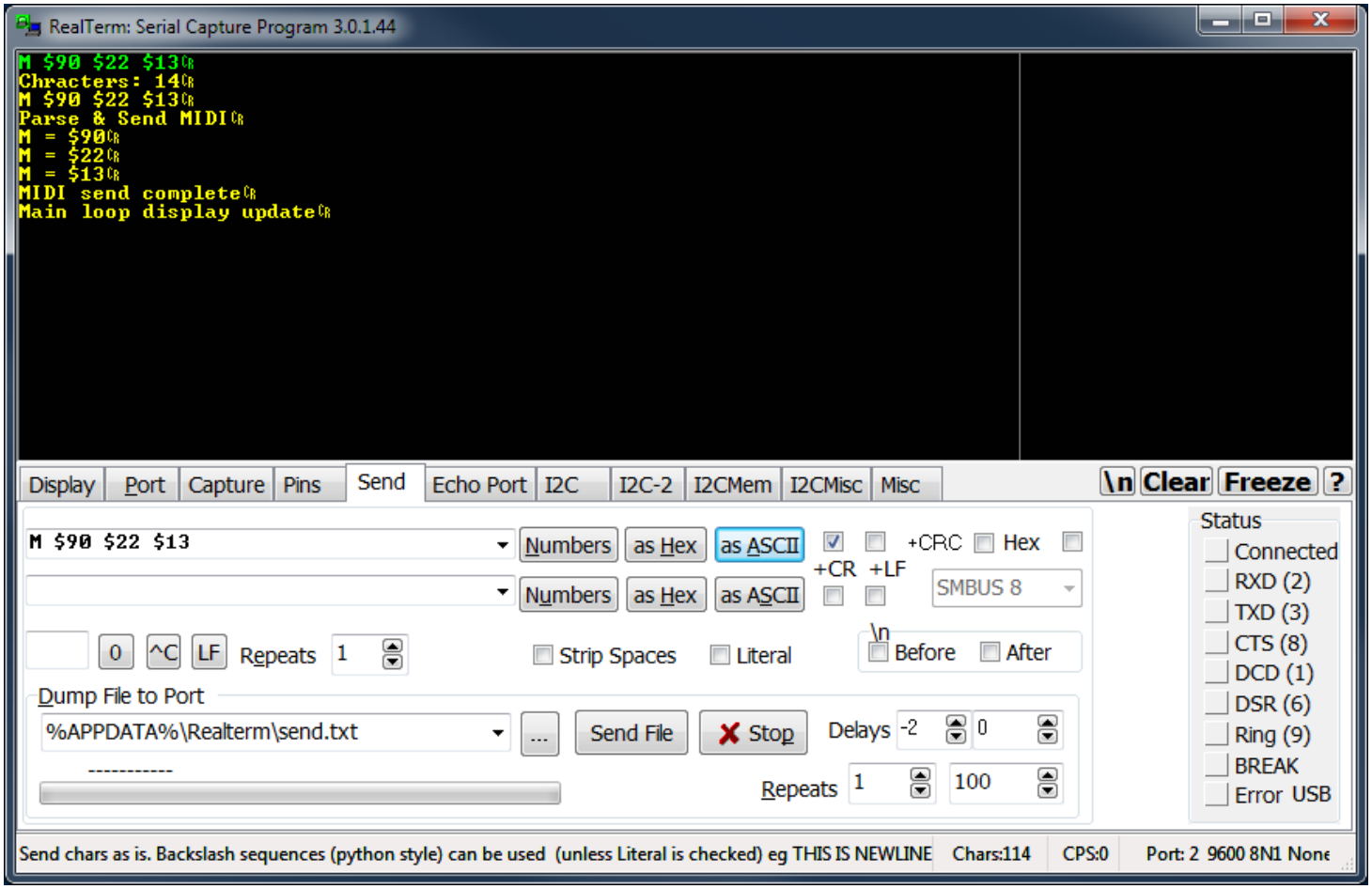
```
\xF0\x01\x02\x03\x04\x05\x06\xF7
```

Sending MIDI to Data to the Decabox

Once the 2-3 byte MIDI messages have been correctly formatted, the next step is to send them to the DecaBox. That syntax looks like this:

```
M $AA $BB $CC...$FF[cr]
- The command starts with a capital M.
- It is followed by single space character ' '.- Then hex values written as strings in ASCII format.
- Command terminates with a carriage return.
Examples, using the same commands listed above:
M $90 $01 $7F[cr]   Note on, MIDI channel 1, note #2, full velocity M $8F $07 $40[cr]   Note off, MIDI channel 16, note #8, 50% velocityM $B1 $03 $10[cr]   MIDI CC #4 (foot controller), ~30% intensity, MIDI channel 2M $F0 $01 $02 $03 $04 $05 $06 $F7[cr]   A short MIDI sysex message. These are framed by [F0 F7] always.
Each of these commands are sent as ASCII STRINGS, whose length varies between 13 and ~30 characters.
```

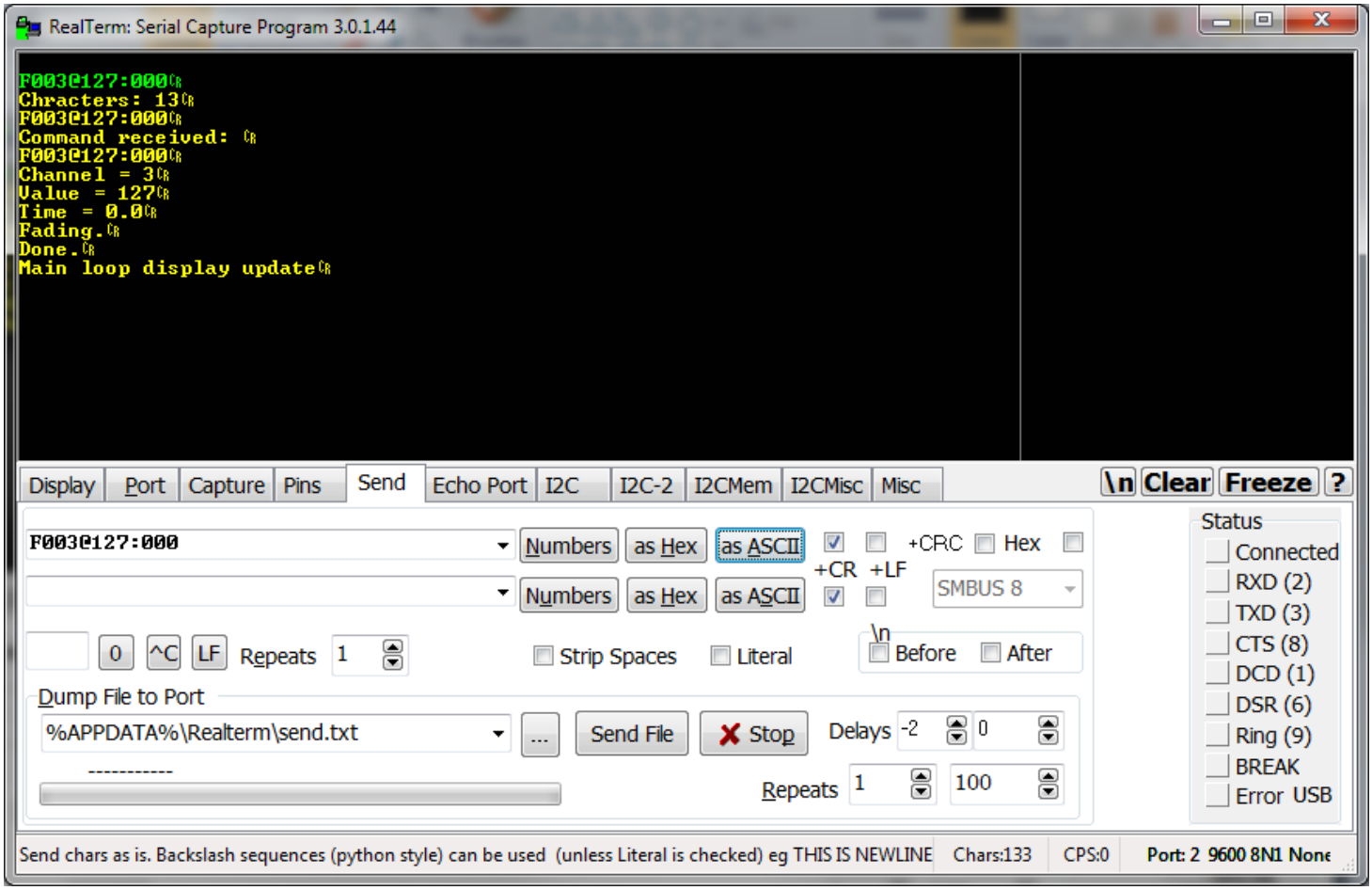
Screenshot. Green is text sent to the box, yellow are replies.


RealTerm: Serial Capture Program 3.0.1.44
M $90 $22 $13
Chracters: 14
M $90 $22 $13
Parse & Send MIDI
M = $90
M = $22
M = $13
MIDI send complete
Main loop display update
Display
Port
Capture
Pins
Send
Echo Port
I2C
I2C-2
I2CMem
I2CMisc
Misc
\n
Clear
Freeze
?
M $90 $22 $13
Numbers
as Hex
as ASCII
+CRC
Hex
+CR
+LF
Numbers
as Hex
as ASCII
SMBUS 8
0
^C
LF
Repeats
1
Strip Spaces
Literal
\n
Before
After
Dump File to Port
%APPDATA%\Realterm\send.txt
...
Send File
Stop
Delays
-2
0
Repeats
1
100
Status
Connected
RXD (2)
TXD (3)
CTS (8)
DCD (1)
DSR (6)
Ring (9)
BREAK
Error
USB
Send chars as is. Backslash sequences (python style) can be used (unless Literal is checked) eg THIS IS NEWLINE
Chars:114
CPS:0
Port: 2 9600 8N1 None

Build a DMX Scene Channel by Channel

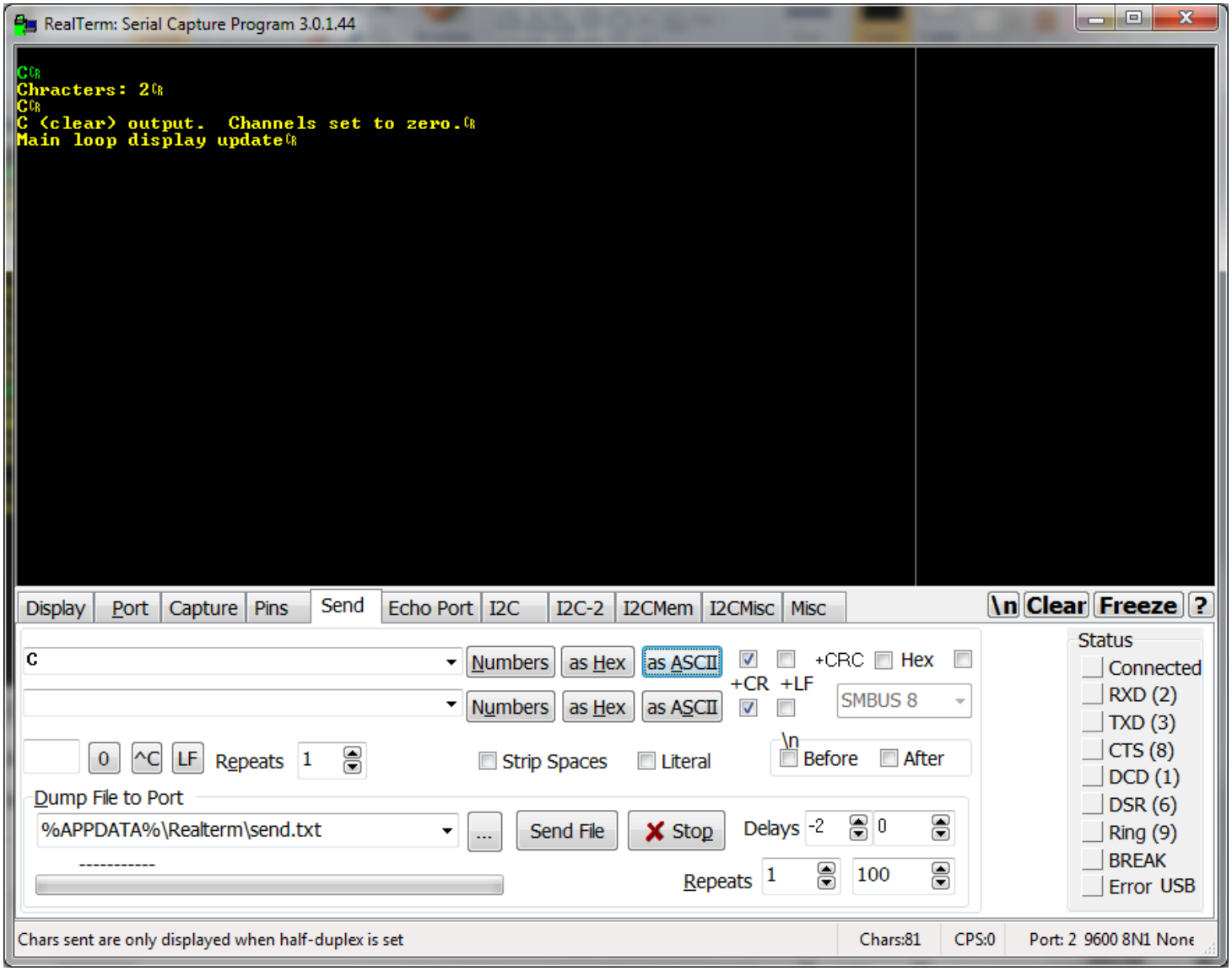
DecaBox can receive RS-232 commands and use them to set specific channels to levels. The syntax looks like this:

```
FXXX@YYY:TTT[cr]
XXX is the DMX channel, range [000 512]. Leading zeros are required.YYY is the channel value, range is [000 255]. Leading zeroes are required.TTT is the time, in tenths of a second, that the internal crossfade
 engine should take to completely process this command. Range is [0 999]  in tenths of a second. 23 -> 2.3 seconds. 999 -> 99.9 seconds.
[cr] is a carriage return.
```



All DMX output channels may be cleared (set to zero) quickly by sending the command

```
C[cr]
C for Clear.
```


RealTerm: Serial Capture Program 3.0.1.44
C
Chracters: 2
C
C (clear) output. Channels set to zero.
Main loop display update
Display
Port
Capture
Pins
Send
Echo Port
I2C
I2C-2
I2CMem
I2CMisc
Misc
\n
Clear
Freeze
?
C
Numbers
as Hex
as ASCII
+CRC
Hex
+CR
+LF
Numbers
as Hex
as ASCII
SMBUS 8
0
^C
LF
Repeats
1
Strip Spaces
Literal
\n
Before
After
Dump File to Port
%APPDATA%\Realterm\send.txt
...
Send File
Stop
Delays
-2
0
Repeats
1
100
Status
Connected
RXD (2)
TXD (3)
CTS (8)
DCD (1)
DSR (6)
Ring (9)
BREAK
Error
USB
Chars sent are only displayed when half-duplex is set
Chars:81
CPS:0
Port: 2 9600 8N1 None

Toggle DMX Passthrough

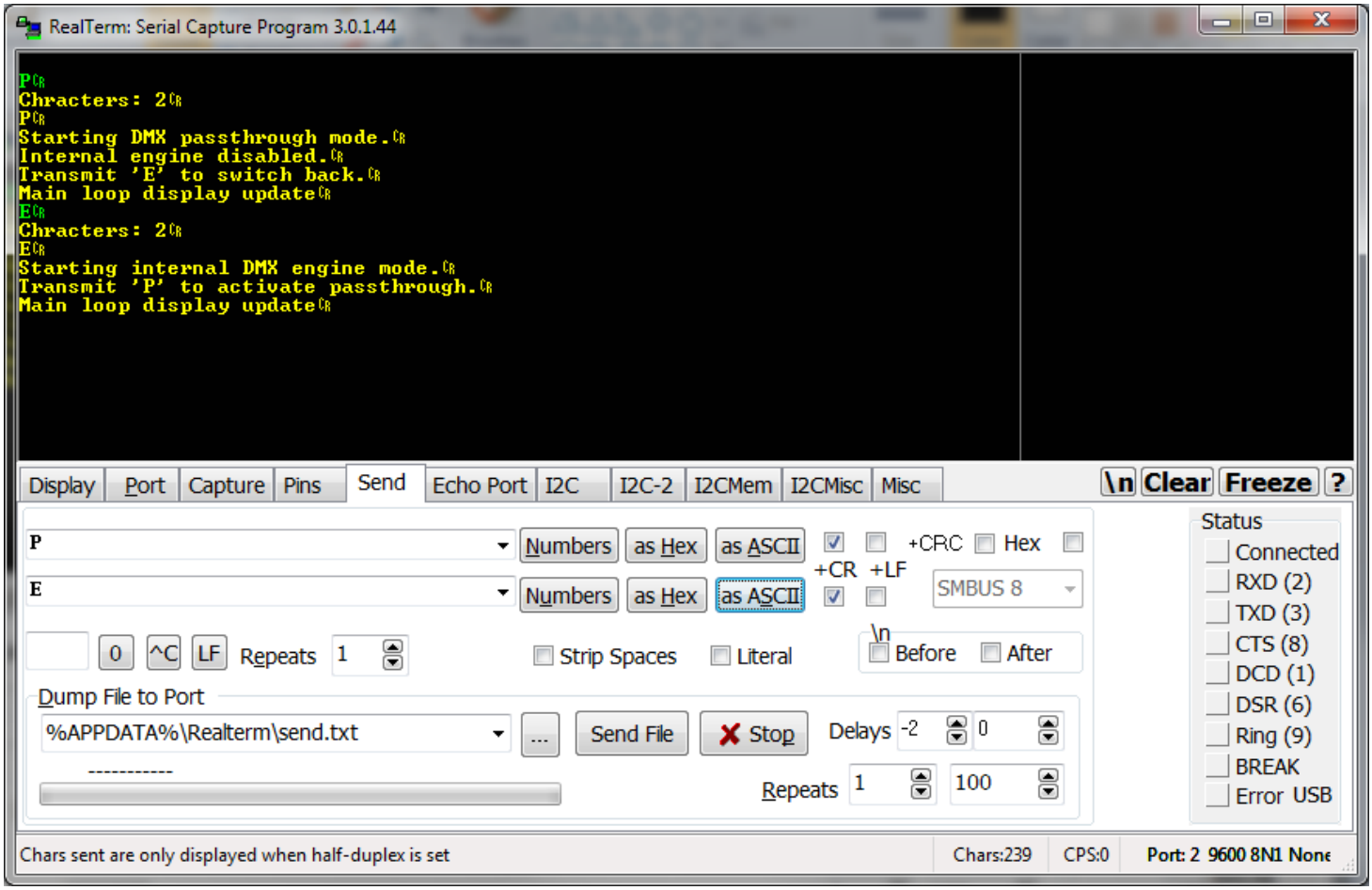
Imagine this scenario:

In an auditorium or small performance space, there is a Crestron / Control4 / AMX control system installed. Touchscreens or switchpanels are mounted near the doorways. On weekends or during large productions, the lighting designer uses a large-format console or PC to run the show. But for rehearsals, cleaning, etc, only a few different lighting scenes may be required. The DecaBox can easily accommodate this, based on its three DMX connections.

- **DMX In** can be connected to any upstream controller, such as a lighting console or USB / DMX interface on a PC.
- **DMX Through** is a passive, parallel copy of DMX input.
- **DMX Out** is connected to our internal processor.

In *passthrough* mode, DMX data is copied from input to output. System lag is ~ 1 DMX frame. It's virtually unnoticeable.

```
P[cr]
... enables passthrough mode.
Data is copied from input to output.
System delay is ~ 1 frame, or 1/40 second.
E[cr]
... tells the DecaBox to ignore
DMX input and instead generate its own scenes,based on serial input, stored data, etc.
```



Thus it can be very straightforward to for in-wall touchscreens, for example, to recall stored DecaBox scenes during the day when unskilled users are present, but for large productions, the lighting crew has complete control.

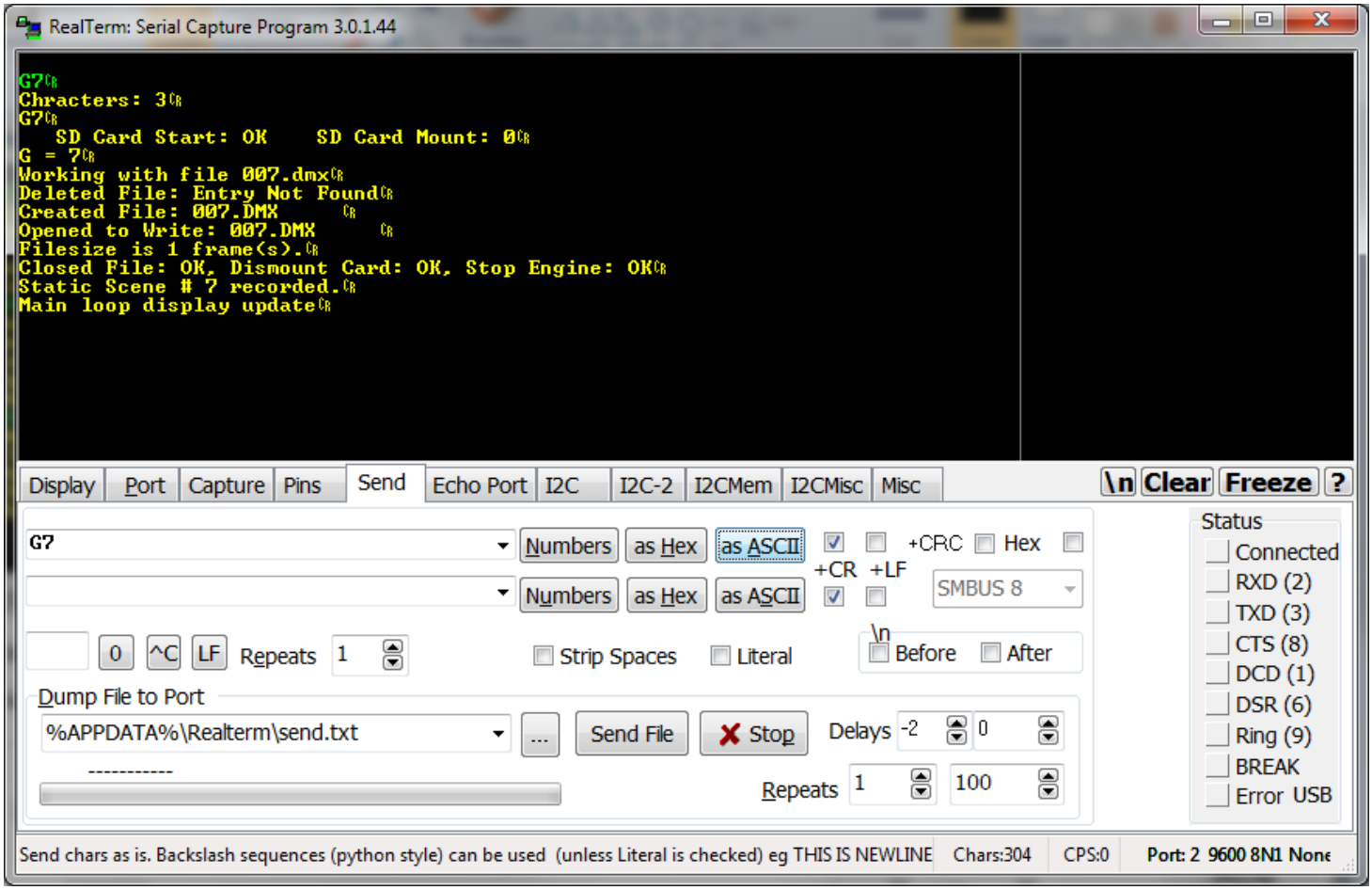
DMX Record & Playback

The DecaBox can *grab* static (one-shot) DMX scenes or *record* continuous sequences at 44 frames per second.

To grab a scene, send the serial command

```
GN[cr]
G is a capital G and mnemonic for grab
N is a decimal number, range is [1 999][cr] is the carriage return
```

A single frame of DMX is stored on the internal memory card. Here’s a screenshot of the process. Again, green commands are generated by the user, and yellow is feedback from the system.



Alternately, dynamic, complicated DMX scenes may be captured at 40 frames per second. The internal storage can hold ~ 9 hours of data.

RN[cr]

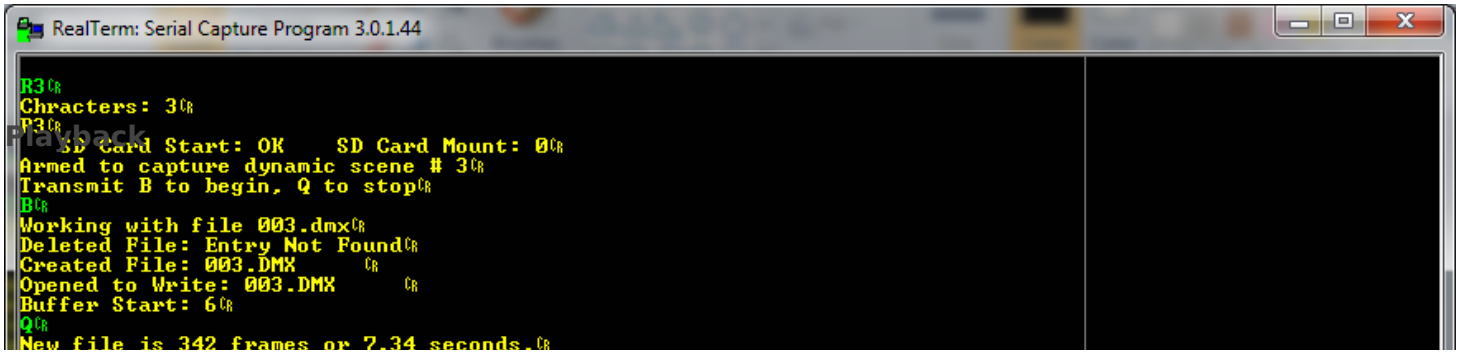
where

Ris a capital R and mnemonic for record

N is a decimal number, range is [1 999]

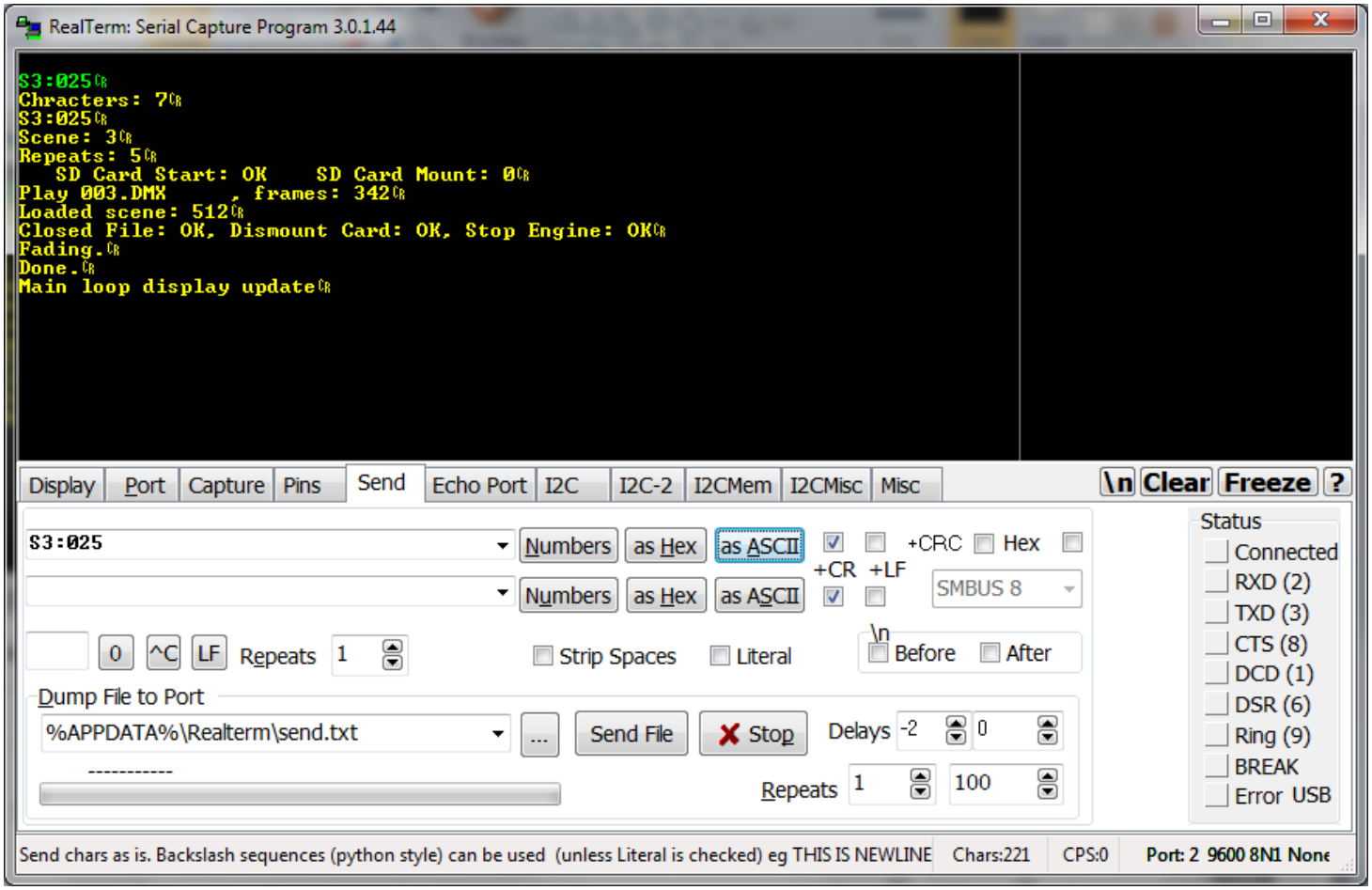
[cr] is a carriage returnThe system creates the file for recording, and then prompts for a ‘start’ trigger.

B[cr] begins the capture and Q[cr] ends it, as shown below.



Once DMX data has been recorded, it's useful to replay it. To that end, both static scenes and dynamic recordings may be called through the serial port. To play a static scene, the syntax is

```
SNNN:TTT[cr]
S is a capital S, and mnemonic for static.NNN is a three digit decimal number, which matches a pre-captured scene
: is a colon
TTT is a crossfade time, in tenths of a second, between the current DMX   output and what's stored in the file. Range is [0 999].
  [cr] is a carriage return
```



To replay a complicated dynamic scene, much like a video clip, the syntax is

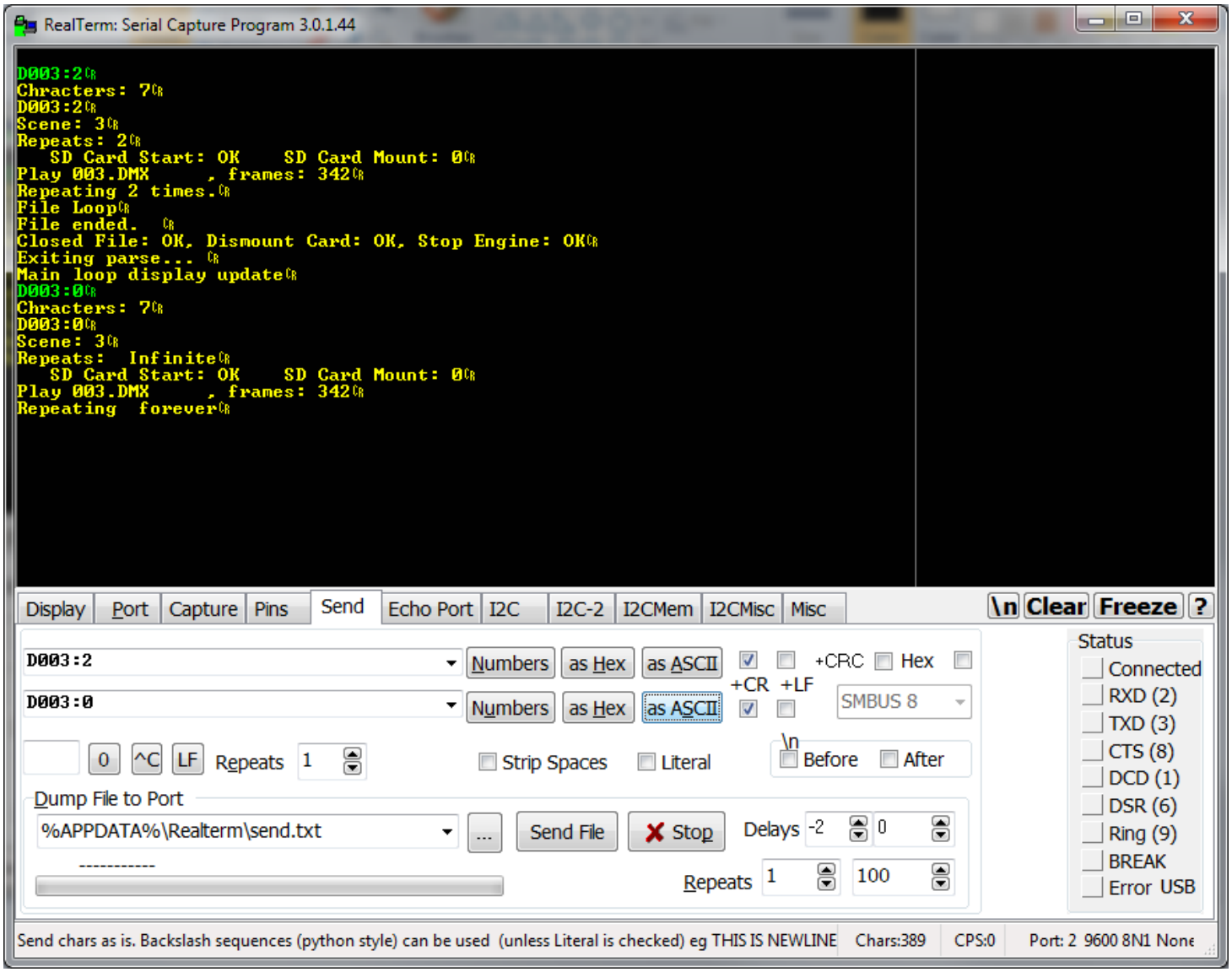
```
DNNN:RRR[cr]
where
D is a capital D and mnemonic for dynamic.NNN is a pre-recorded scene number, three digits,
```

```
LEADING ZEROS REQUIRED.

: is a colon

RRR is the number of times a file repeats.  Use 000 for infinite loop.

[cr] is a carriage return.  Decimal 13 or 0x0D
```



Purchasing

If you've landed here via a web search, don't already own one of these systems, but would like to add one to your collection, our online store is here:

www.response-box.com/gear/shop

And our main site is here, which includes links to distributors, etc:

www.response-box.com/gear