ETC/LMI Serial Button Protocol Definition

Electronic Theatre Controls, Inc.

Last Revision November 9, 1990
for Software Release 1.84

Introduction

The ETC Serial Button Protocol provides a means by which a host device (e.g. a personal computer) can send operational commands to an ETC Expression, Impression, Insight or Concept 500 lighting control console. For example, the Serial Button Protocol makes it possible to use a PC remotely to control a console that is configured with no face panel.

Communciation is bidirectional, via an RS-232 serial link at 9600 bps, with 8 data bits, 1 stop bit. no parity. The host device uses a distinct serial port for each console slaved to it.

Communication is packet-oriented, with each packet consisting of a list of up to 32 opcodes followed by a termination code. Certain opcodes require one or more arguments – such arguments follow immediately after the opcode.

In what follows, word refers to a 2-byte integer value. Word values are transmitted least significant byte first.

Escape Seauences

Each packet is terminated with a byte value of 255 (decimal). To distinguish between a terminator and other occurrences of 255 in the packet data, escape sequences are employed in the standard manner.

That is, an escape character (byte) is defined; it is 27 (decimal). Each non-terminator instance of 255 in a packet is preceded by an escape character. Likewise, each non-escape byte value of 27 in a packet is preceded by an escape character.

Escape characters are inserted into the packet as needed by the sending device when a packet is transmitted and are stripped out by the receiving device when the packet is received.

Escape characters are used only for host-to-console communications. They are not used in the console-to-host Status Message (see below).

Host-to-Console Operational Opodes

Up to 32 opcodes may be sent in a packet. The only exception is opcode 1000, which must be sent one-per-packet. The opcodes are word values, as are their arguments, if any.

Many of the opcodes correspond simply to buttons on the fade panel of the lighting control console. For that reason, they will be referred to as 'button' in this document.

The following opcodes have no arguments:

Opcode	Definition
1	Up Arrow
2	Left Arrow
3	SETUP
4	AUTOLOAD
5	PATCH
6	STYLE
7	TRACKSHEET
8	CUE SELECT
9	FADER
10	STEP
11	EXPAND
12	Right Arrow
13	BLACK OUT
14	Down Arrow
15	INHIB SUB
18	HOLD A/B
20	HOLD C/D
21	REC
22	GO A/B
23	LINK
24	TRACK
25	GO C/D
26	M1
27	M2
28	M3
29	M4
30	M5
33	REL
34	EXCEPT
35	SOLO
36	TIME
37	DIM
38	TYPE
39	PAGE
40	SUB
41	CUE
42	7
43	BLIND
44	8
45	STAGE
46	4
47	CHAN
48	5
49	GRP
50	1
51	AT
52	2
53	FULL
54	0
55	CLEAR
56	-
57	ENTER
58	+

59	AND
60	
61	THRU
62	33
63	9
64	6
89	M*
90	MACRO WAIT
91	ENTER MACRO
93	CLEAR A/B
94	CLEAR C/D
95	BACK
100	FLASH
104	HELP
109	Clear the System

1003 Update digitzer region definitions (See opcode 1002, below). After sending new digitizer cell definitions via a sequence of opcode 1002 packets, send an opcode 1003 to cause the console to 'digest' the new cell definitions. Cells that have nto been modified, i.e. cells for which no opcode 1002 packet was sent, retain their current definitions.

The following opcodes have 1 argument:

Opcode	Definition
106	Submaster bump
107	Clear-sub (a conditional submaster bump: if the submaster is currently fading up or waiting to begin its downfade, fade it out. Otherwise, do nothing. i.e., net effect is to fade the submaster out).
108	On-sub (a conditional submaster bump: if the submaster is not already fading up or waiting to begin its downfade, fade it up. Otherwise, do nothing. i.e., net effect is to fade the submaster up.)
157	Grandmaster pot level (for console with no face panel)
161-164	A,B,C,D fader pot levels, respectively (for console with no face panel)
170	Fader wheel

Recall that arguments are word values.

For opcodes 106-108, the argument is a submaster number, 1-108.

For opcodes 157 and 161-164 the argument is a value in the range 0-255 specifying the pot level.

For opcode 170, the argument is a value in the range 0 to 200, specifying a biased number of 'ticks.' A tick is the smallest wheel movement detectable, analogous to a 'mickey' in mouse nomenclature. The values are biased by adding 100 to the number of ticks, so that 0 corresponds to -100 ticks, 100 corresponds to 0 ticks, and 200 corresponds to 100 ticks. This biasing *is* done to avoid negative values, since -17 and -19 are treated specially by the console (see the section 'XON/OFF Pacing', below).

The following opcodes have 2 arguments:

Opcode	Definition
92	Alphanumeric keystroke (for consoles with alphanumeric keyboard option) Arguments: (ASCII code, scan code)
1004	Submaster pot level (for console with no face panel) Arguments: (submaster number, pot level) Pot level is in the range of 0-255.

The following opcode has 3 arguments:

Opcode	Definition
1002	Assign a region to a cell of digitizer region space (See opcode 1003, above). Arguments: (Region number, X coord, Y coord) Coordinate system: 0.1 inch grid, lower left cell = (0,0) i.e. to program digitizer regions, send a sequence of packets using opcode 1002 to define the region assignment for each 0.1 inch cell to be redefined in the user-programmable portion of the digitizer, then send opcode 1003 to tell the console to do the internal processing required to set up the new regions in memory. Cells that are not modified, i.e. cells for which no opcode 1002 is sent, retain their current region assignment. To clear a cell, assign it to region 0. The macro associated with a given region can then be programmed by sending the button hits you would use to program the macro from the console face panel.

Terminator

As noted above, each packet is terminated by a byte value equal to 255 decimal.

Sample Packet

The following stream of bytes (decimal) sent host-to-console starts CUE 25 on the A/B fader, triggers Macro 2, moves the fader wheel -1 tick, and sets the Grandmaster to 50%:

41 0 52 0 **48** 0 22 0 27 27 0 170 0 27 255 27 255 157 0 127 0 255

Note particularly the use of escape sequences.

Host-to-Console Status Request Opcode

The following opcode allows the host to request status information from the console.

Opcode Definition

1000 Status Request (no arguments)

The console replies with a Status Message (see below). One special limitation applies to opcode 1000:

A packet containing opcode 1000 is not permitted to contain any other opcodes.

Console-to-Host Oucode

Opcode Definition

1001 Status Message **(1** argument)

The status message is sent in response to opcode 1000. It is also sent whenever the console error status changes.

Argument: 16 l-bit condition flags: flag = 1 means the specified condition has occurred.

Bit 0 - battery memory error
Bit **1** - disk error
Bit **2** - printer error
Bit 3 - communications buffer overflow
Bit **4** - host must wait for XON before resuming transmission
Bit 5 - a macro is active
Bit 6 - console configured with no face panel (sent only when console boots up)
Bit 7 - fae panel communications timeout

Bits **8** - 15 - reserved for future use

The Status Message is guaranteed never to have XON or XOFF bytes (see below) interleaved within it. As a precaution, therefore, each Status Message is routinely preceded by an XOFF and followed by XON.

Likewise, Escape sequences are not employed within the Status Message.

Sending Macro Commands from the Host

Macros on ETC consoles have the following property: starting a macro while another macro is running terminates the macro that was running.

If the host wants to start a macro, but does not want to **risk** terminating an active macro, it should first send a Status Request packet to see if a macro **is** active (see the Status Message definition, above.) If the Status Message reply packet from the console indicates a macro is active, the host should re-send the Status Request and should continue to do so until the Status Message reply indicates no macro **is** active.

Note that is is possible to define a macro that runs forever – e.g. a macro that executes itself. It is the responsibility of the user to ensure that a deadlock condition does not arise, for example by stipulating that no 'infinite loop' macros will be used in the console when a host will be sending macro commands, or by building a timeout into the host when it is waiting for the console to finish its currently executing macro.

Note also the distinction between a macro that 'runs forever' and a macro that initiates some other action that 'runs forever'. In particular, suppose a macro is defined that does nothing but initiate a submaster bump with an infinite wait time. The macro is active only so long as it takes to initiate the bump. This is **not** an example of a macro that 'runs forever' – no deadlock consideration apply.

Miscellaneous Comments

Opcodes not defined above are reserved for future expansion.

The console face panel ordinarily remains 'live' while the protocol is in use. If buttons transmitted from the host device and buttons sent from the console's face panel are interleaved, the operational results will be unpredictable. This problem is alleviated partially by the following stipulation: console software gives a higher priority to buttons transmitted from a host device. This means that a sequence of buttons sent in a single packet will necessarily be serviced sequentially.

Note, however, that there is nothing to prevent a sequence of buttons entered at the face panel from being interrupted by buttons sent from the host device. It is the responsibility of the user to ensure that this does not interfere with the operation of the console. This may be done by carefully partitioning operational tasks between the face panel and the host device, or by restricting use of the face panel to operational phases during which the host device is known to be inactive.

The serial button protocol can be used to set pot levels only on consoles that are configured without a face panel. If the console has a face panel, pot levels transmitted from the host device will be ignored. Fader wheel movements, however, will be processed by the console in any case.

XON/OFF Pacing

The console uses XON/XOFF codes to pace transmissions from the host. The host does not pace the console. XON's and XOFF's from the host are harmless; they are ignored by the console.

If the console's input buffer becomes nearly full, the console sends an XOFF byte to the host. The XOFF byte value is 19 decimal. It is sent as a single bute, not as a packet with opcode and terminator. The console sends an additional XOFF for each byte received until the host stops transmitting.

When the console has emptied most *of* its buffer, it sends an XON byte to the host. The XON byte value is **17** decimal. Like the XOFF, the XON is sent as a single byte, not as a packet with opcode and terminator.

The host stops transmitting when it receives an XOFF. It resumes transmitting when it receives an XON. If no XON is sent within a reasonable time period, the host should send a Status Request message. Then if the host does not receive a Status Message in reply within a few seconds, it may assume communications with the console have been disrupted. That is, the console guarantees that replies to Status Requests will be sent promptly even if the console is in the midst of doing something else.

Note that a Status Request message (the only console-to-host message) is guaranteed never to have XON/XOFF bytes embedded within it. **As** a precaution, each Status Message is routinely preceded by an XOFF and followed by XON.

A minimal implementation of host software may ignore 'XON/XOFF pacing provided the host adheres to the following three requirements:

- First, the host must not send packets so quickly that the console cannot keep up. (To get a sense of how fast is too fast, see the DEMO program, described below.)

- Second, the host must not send packets while the console is busy doing a disk operation. In particular, this means that a command that initiates a disk operation must be the last (or only) command in its packet.

- Third, the host must delay at least a tenth of a second after sending a command that causes the console to switch screen modes. Such commands include: STAGE, BLIND, FADER, TRACKSHEET, PATCH, SETUP, and EXPAND. In particular, it follows that any such command must be the last (or only) command in its packet.

Since XON (17 decimal) and XOFF (19 decimal) have special meanings in the protocol, it is necessary to distinguish them from values of 17 or 19 that may occur as arguments (e.g. to specify submaster 17 or 19). Argument values of 17 or 19 are treated as a special case:

IMPORTANT All argument values of 17 and 19 must be sent to the console as -17 and -19, respectively. The console always converts -17 to 17 and -19 to 19.

Since arguments are word values, sent least significant byte first, and since non-terminator occurrences of 255 are escaped:

-17 is sent as: 239 27 255 (decimal)
and
-19 is sent as: 237 27 255 (decimal)

Latency

A packet is not serviced until its terminator byte has been received. This introduces a slight delay (roughly N millisec., where N is the number of bytes in the packet) before the first opcode in the packet can be serviced by the console in the best case.

Worst cases are possible. The console may already be servicing another button which must complete before the packet's opcodes can be serviced. Most buttons take only about .05 secs. to complete, but some – disk and printer operations being the worst cases – take longer.

Since buttons received from the host are given priority, at most one button must be serviced before the next button from the host is attended to. This means that host transmitted buttons will ordinarily receive prompt service. If large numbers of buttons are sent by the host in rapid succession, however, buttons sent later will have to wait their turn. This means that the host may want to prioritize among the various operations in transmits.

Setting Up the Console

To use the Serial Button Protocol with your Expression, Impression, Insight or Concept 500 lighting control console, the console must have sofrware release 1.60 or later. The details of the protocol are described in this document revision apply specifically to the software release noted on the title page.

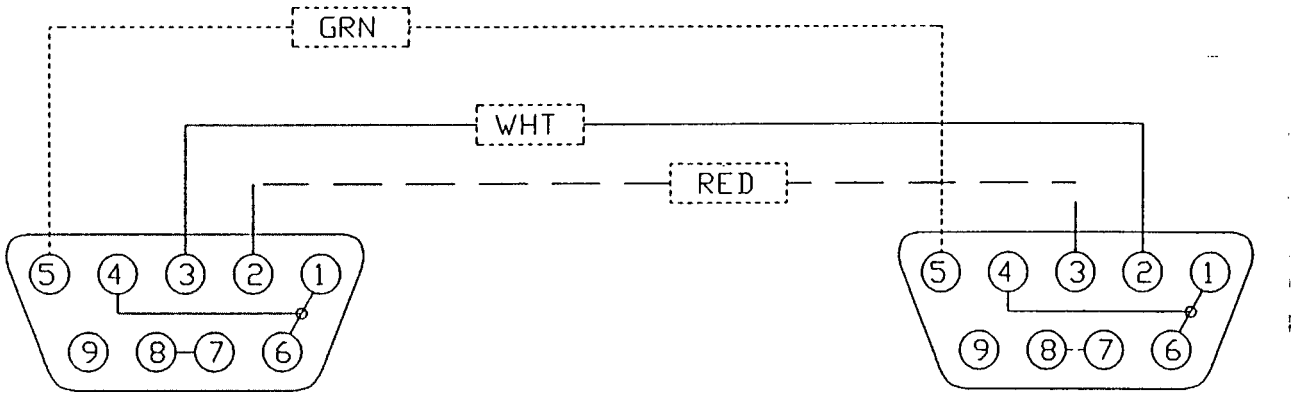
Connect the Serial Button Protocol cable between a serial port on your host device and the serial printer port on the back of your console.

In the System Settings menu on your lighting console, select the "Serial/Parallel Printer' entry and set your console for a parallel printer. This allows the Serial Button Protocol to use the serial port.

Having selected the parallel printer, turn your console's power off and back on to reboot it. When the console completes the boot sequence it is ready to receive Serial Button Protocol commands.

BELDON 9503 CABLE

6 FEET



DB-9 FEMALE
W/ STRAIN RELIEF

DB-9 FEMALE
W/ STRAIN RELIEF

PINOUT:

2		2	Data In
3		3	Data Out
5	——	5	Gnd
1,486	nc	1,486	Console +12v
788	nc	788	RTS&CTS

INSTALLATION:

- Cable is connected to "Serial Printer" connector on console.
- Connectors are wired for PC AT style serial ports,

SERIAL BUTTON PROTOCOL INTERCONNECT CABLE ELECTRONIC THEATRE CONTROLS 10-10-88	EXPN-481 PAGE 1 OF 1