

## **CPU EVENT PROGRAMMING**

Version 2.2 12/31/97

### **TABLE OF CONTENTS**

- EVENT TYPES
- CPU EVENT SYNTAX SUMMARY
- ON: EVENTS
- EVENT ENTRY AND EDITING
- LOADING AND SAVING EVENTS
- REORGANIZING EVENTS
- MERGING EVENT FILES
- MARKING EVENTS
- EVENT LOCKING SYSTEM PRELIMINARY
- MACRO ENTRY AND EDITING
- SONY LDP CODE CHART AND APPLICATION
- PIONEER 8000 CODE CHART
- SB - SOUND SYNC RECORDER/EDITOR FOR SOUND FILES
- NOTES AND UPDATES

This document is intended for the installation and support of Show Control Systems supplied by Triad Productions, Inc. and contains trade secret information regarding proprietary inventions and processes. This information may not be duplicated or revealed in any form without express permission of Triad Productions, Inc.

© 1989-1999 William J. Synhorst/Triad Productions, Inc. All rights reserved.

We believe this information to be correct and accurate at the time of printing. All specifications are subject to change. Triad will not be responsible for any damage related to any use of this equipment or documentation. Please report errors or omissions to Triad Productions.



The Events sub-system allows direct access and control of all digital and analog channels, serial communications channels for source deck, projector, or laser disc control, in addition to lighting and other functions on a predictable show time basis, without necessarily having to program in real time (as for animation). Once an event script has been entered, all steps will occur at the exact point of the show time code specified, but if the code is advanced (tape wound fast forward or spliced), all unprocessed events will be executed up to the current show time. This is very useful for interlocked control sequences that could otherwise miss a pulse, as well as lighting and other systems which must receive all cues to stay in sync.

A powerful feature is the ability to define **subroutines**, or complex lines of instructions, which may be invoked (at a specified time) using a simple reference number. It is also very easy to move or offset events, by simply changing the SMPTE time at which occurs, then Reorganize the events file into sequential order, which can be considerably faster and easier than assigning a channel to a switch or pot and reprogramming it with show time code.

When using the integral Lighting programming subsystem, light cue times are automatically added and sorted into the EVENT list. Only the first time code reference will be entered, and only if the time for the cue is greater than 00:00.00. Note that the events file is saved independently of the lighting cues, therefore BOTH files must be saved to disk if changes are made in the Lighting system.

**Note: Lighting cue #0 is NOT automatically added to EVENTS. Also, it is possible to create multiple calls to the same lighting cue as additional Events.**

New features have been added which allow conditional tests to be made using the SMPTE time, counters, and other variables. This allows branching, looping, and invoking additional subroutines by a timed event or an external, asynchronous closure.

Event **subroutines** may be created for frequent or repetitive operations, or for events that must occur independent of the absolute show time code.

Event subroutines may also be "armed" for asynchronous or external triggers, or after preset delay times. This feature, combined with control of up to 32 asynchronous, simultaneous animation data macros allows the system to actually control virtually any number of sub-functions or sub-shows concurrently in frame-locked but relative sync. Triggers may be activated by remote contact closures, ASCII serial messages, or keyboard function keys assigned to a particular subroutine/sequence.

## EVENT TYPES

Note: Most values are represented in decimal, and start with card/channel '1' (not '0'). Hex numbers may be entered using a "\$" prefix, but in most cases will be re-displayed in decimal!

The current events that are defined are as follows:

1. **marker** Identifies a specific show time code M.arked while playing through the show. The marker may then be changed into any regular event, and parameters added. Pressing the "M" key will flag the current time as a marker at the **end** of the timed events queue when E.xecuting with show time code.
2. **DigOn** (Digital On) - Turns on the digital function specified by the following character (card) [1-16] and the channel to turn on [1-32].
3. **DigOff** (Digital Off) - Turns off a specified card/channel.
4. **Pulse** Sets up a half second digital pulse (on-wait-off) to the card (1-16) and channel (1-32) specified. CAUTION: A pulse will only execute properly when one of the SYNC modes (SMPTE, internal, etc.) is in effect! Use a DigOn/Wait/DigOff sequence to generate a pulse in ON: Event sequences.

NOTES: In the Synthesis system, 16, 24, and 32 subchannel digital outputs are supported. The number of subchannels interpreted by an I/O frame is determined by a parameter in the specific card frame (i.e. SCU/LDC, CTU, DTU or STU). If the number of subchannels is set to 24 (i.e. for TC-124 Digital Output cards), any reference to a 16 channel card (TC-100, TC-316, or TC-3161) must be offset by eight (8); so that function 9 corresponds to the first subchannel of a 16 channel card. Likewise, if the number of subchannels is set to 32, an offset of 16 may be required. In current systems, two physical 16 channel cards are strapped to map into one logical digital channel, with subchannels 1-16 on the first card and 17-32 on the second. Refer to the I/O map associated with the particular system and configuration.

It is possible to use "X" variables for both the card and subchannel numbers. First establish the X array variables for each. Then use function 26 (SetXY) to set scalar variables X and Y to be based on the array index for each variable. To use the value of X as the card/channel number, use PARM1=255; to use the value of Y for the subchannel, use PARM2=255.

```

S002. 001  Xp1=p2  001 005      (sets X(1) = 5)
S002. 002  xp1=p2  002 012      (sets X(2) = 12)
...
S003. 001  Set XY  001 002      (sets X=X(1), Y=X(2))
S003. 002  Di g0n  255 255      (turns on channel 5, subchannel 12).
```

5. **AnValu** (Analog Value) - Forces the indicated channel in PARM1 [1-512] to the value specified in the second parameter (0-255). An "X" variable may be used as the channel by specifying 513 as the channel in PARM1. A "Y" variable may be used as the value by specifying 513 as the value in PARM2. More information on X variables may be found later in this document.

The assigned NAME for an analog or digital channel is displayed for events referencing an analog or digital function once entered. Using the up and down arrow keys, the channels and associated names may be scanned while entering the parameter(s) for analog or digital events. Channel names may be entered or edited under A.ssign - N.ames or from a hard copy listing.

- 6. System** Allows special system functions to be executed from within an event. The value of the first parameter indicates the system function:

VALUE	FUNCTION
0	Clear SYS commands (Yak/Rnd/Etc.) (DOT)
1	RESET - Execute the RESET macro, set time to 00:00
2	XMIT - Force update of all digital/analog outputs to the I/O frames (if not BLIND)
3	SET SMPTE
4	SET TRACE
5	SET PILOT
6	{ Encode to data-on-tape
7	{ but do not process...
8	{ EOT ENBLE
9	{ END ENBLE
10	STUFF Keyboard Buffer with characters in ASCII Macro in PARM2. This allows virtually any function or series of steps to be performed without an operator. As an example, with a simple macro string of "e\" in the ON:INIT events, the system will automatically begin in execute mode when the program is started.
11-14	Pulse commands in the TC-3500/TC-3550 stand-alone systems ONLY
15-16	Latch on commands in the TC-3500/TC-3550 stand-alone systems ONLY
17-21	Latch off commands in the TC-3500/TC-3550 stand-alone systems ONLY

The following SYSTEM events are only meaningful within the Synthesis system running on a host PC computer:

VALUE	FUNCTION
31	Load the CUE file named in ASCII macro (PARM2)
32	Load the DEFAULT.DFL file named in (PARM2)
33	Load the EVENT.EVT file in macro (PARM2)
34	Load the LIGHT.LGT file as specified in (PARM2)
35	Load a MACROS.MCS animation file as specified in (PARM2).  Note: Multiple .MCS files may be loaded into EMS memory up to the maximum size of EMS memory available with a maximum of 32 macro IDs. Thus, a separate EVENTS file may be created as a "script" file to build a complex MACRO file from a series of individual cue files.  The file name is entered as an ASCII Macro string using DOS file Name conventions. When using the file load event types, be careful NOT to use the "\" comment delimiter, as DOS will interpret it as a path specifier.
45	Lighting blind. PARM2<>0 inhibits lighting cues.
46	Events blind <> 0 will inhibit execution of timed events.
47	Event -> cue conversion flag. PARM2 specifies the type(s) of events that will be flagged for conversion into real-time animation data. Digital and analog channels that are to be converted must be flagged in the current CHANNEL SET. bit 7 (128) = 1 = convert digitals bit 6 (128) = 1 = convert digbytes bit 5 (32) = 1 = convert analogs bit 4 (16) = 1 = convert subroutine calls
48	Keyboard lockout, set by PARM2<>0, cleared by PARM2=0
49	Cue Execution lockout, set by PARM2<>0, cleared by PARM2=0
60	As of 06/27/97, forces the system clock to reset from the hardware clock at midnight (if in EXECUTE).
80	Inhibit sending COM1: event strings

81	Enable COM1: events (default)
82	Inhibit sending COM2: event strings
83	Enable COM2: events (default)

The last executed SYS number will be displayed in the lower left corner of the EVENTS execution window.

These events are used primarily for encoding control commands for data-on-tape applications.

- 7. Pause** Beginning with release 96.11, PAUSE now acts as a delay within a subroutine between "sorting" or "point" numbers with PARM1 being the number of SECONDS to PAUSE and PARM2 being the number of FRAMES. This allows for complex delay structures to be built **within** a subroutine (no need for "bucket brigade" delay subroutines). Use an ArmFrm (33) with a PARM1 of the subroutine to be killed and PARM2 of 0 to kill a (looping) subroutine with pauses embedded. (ArmFrm 0 only works on the CPU.)

```
S001.001 Pulse 005 001
S001.001 Pause 010 015
S001.002 Pulse 005 002
```

In this example Digital output 5:1 would pulse and 10.5 seconds later 5:2 would pulse.

- 8. E.O.T.** Only available in SCU Events.
- 9. EndEnb** Only available in SCU Events.
- 10. Com1:** Allows a character or macro to be sent to the Triad TC-500/750S SMPTE Reader/Controller, via the COM1: port. If the second parameter is "0", the corresponding ASCII character in PARM1 will be sent as a command code to the TC-500. To simplify the conversion to decimal, you may enter the ASCII character to send by entering \$ (hex) " (literal) c (any character) to perform an automatic translation. If the second parameter is not 0, it is used as a pointer to the **macro** string to send (see below).
- 11. Com2:** COM2: behaves exactly like the COM1: command, in that a non-zero second parameter is used to send a **macro string**, otherwise the first parameter is sent as an ASCII character to the COM2: serial port.
- 12. QCom1:** Queues a message to the COM1: port that will be dispatched at two characters per frame, regardless of the baud rate.
- 13. Com3:** COM3: behaves exactly like the COM1: command, in that a non-zero second parameter is used to send a **macro string**, otherwise the first parameter is sent as an ASCII character to the COM3: serial port.
- 14. Com4:** COM4: behaves exactly like the COM1: command, in that a non-zero second parameter is used to send a **macro string**, otherwise the first parameter is sent as an ASCII character to the COM4: serial port.
- 15. XCom:** This code is used to enter events used by the TC-3550/3518 extended serial communications system. The first parameter is a bitmap of the serial port(s) to use, and the second parameter is the MACRO ID to send. Refer to the EVENTS description specifically for the TC-3550 system.

On the Synthesis system, XCOM has quite a different interpretation. XCOM is used to indirectly link to the serial COM ports on LDC/SCU I/O frames. The first parm determines the primary COM port on the Synthesis PC to use and the target port on

the I/O frame, the second parm is the ASCII macro to send. Single characters (in PARM1) may not be used, and a message may only be sent to ONE target port at a time.

10 - Send to DATA port on TC-3550 via PC COM1: port  
11-18 - Send to TC-3518 serial port 1-8 via PC COM1: port

20 - Send to DATA port on TC-3550 via PC COM2: port  
21-28 - Send to TC-3518 serial port 1-8 via PC COM2: port

The message is lead with an F0..F8 header to select the port at the I/O frame, and followed with a \$FF trailer to end the packet, and directed to the appropriate port on the host PC. There is currently no loop-back or return messages, other than triggered events that are associated with characters that come in either COM1: or COM2: on the host PC. Note that the first parm is NOT a bitmap as in the LDC/SCU version, and only one port may be accessed at a time.

## 16. Disply

Displays the text of the specified **macro** number in PARM2, based on color and position information in PARM1. The high nibble (in \$hex) controls the color, and the low nibble controls the display line, from 0...15. Specifying a null macro string (PARM2=0) will open a "message" window, with enough room for 16 lines of display data, which may be displayed and formatted controlled by other display message events.

*PARM1 is displayed in Hex. Please refer to the Reference section of this manual for the color and position information to go in PARM1.*

*\*Note: Only 62 ASCII macros are available in an LDC/BART.*

## 17. APatch

Allows dynamic reassignment of the physical **ANALOG** output channel number (an actual D/A converter or DMX channel from 1-512) from the logical channel number used during programming. Normally, the logical and physical channels are assumed to be assigned one-for-one and no repatching is required. To install a patch, use this event #, followed by the PHYSICAL OUTPUT channel (TO) and then the LOGICAL channel number (FROM).

00:12.13 Patch 003 001 } devices connected to outputs  
00:12.14 Patch 004 001 } 3 and 4 will follow channel 1.

Note that multiple outputs may follow a single programmed channel, simplifying programming while reducing the number of cues needed. It is further possible to reassign patches at any point in the show. Patching effects lighting and regular animation analog channels. To see the current patch bay, use the P.atch option from the A.ssign menu. The output (physical) channels are shown in the green (odd) columns; the source (logical) in blue (even) columns.

A R.eset will restore all patches to their default assignments, as defined in file "DEFAULT.DFL". (Refer to the ASSIGNS/PATCH documentation.)

## 18. AutoSt

Engages an automatic restart cycle time, based on the minutes (first parm) and seconds (second parm) for this event. Once activated, a window will open, and count down the time remaining before an ON: START event will be processed. A CYCLE event would normally be programmed only as a part of an ON: END or ON: EOT sequence.

## 19. Parall

Redirects the character in PARM1 or if zero, the macro text specified in PARM2 to the parallel (printer) port. Intended for debugging applications, this feature may also be used for logging or parallel control applications. DO NOT USE if the data transmitter or Show Programming Console is already connected through the parallel port!

## X VARIABLES

Beginning with release 6.8, up to 64 "X" variables are permitted. In most cases, the X is selected by PARM1 (from 0-63); for the conditional branches, X is the last value modified or SET by one of the X event types.

On the Synthesis system, X variables have been extended to word variables, meaning that the value may be set from 0 to 512 and then some. Analog channels up to 512 can be accessed (513 to specify an X wildcard), and ARMFRM and ARMSEC may be extended beyond 255. Note that the TC-3500/TC-3550/BART systems will still only use byte (8 bit) values, and that many parameters are not valid beyond their normal bounds, i.e. 255.

**CAUTION:** When a subroutine, armed event, or delay-triggered subroutine occur, the current X value is saved prior to the call, and restored afterward. This is to preserve the X value should an asynchronous event happen between setting and testing "X". If in doubt, put a "SETXY" (26) event code just prior to one of the test/branch events.

Error messages are given if an undefined subroutine is called, or if you attempt to call a subroutine from within a subroutine (nesting No-No). In the future, nested routines may be allowed.

**20. Xp1=p2** Sets variable (counter or flag) X[PARM1] to the absolute value (0-255) of the second parameter. The scalar "X" variable used for testing or modification is also set.

**By setting the high bit of the first parameter to '1' (128+ X element), the second parameter may be one of the variables defined in the following table.**

P2		VALUE	DESCRIPTION
0-63	X Array variable		Allows a variable to be copied
128			
129	XX		Current X
130	XY		Current Y
131	HOURS	0-23	Current Clock Source Time Code
132	MINUTES	0-59	Current Clock
133	SECONDS	0-59	Current Clock
134	FRAMES	0-29	Current Clock
135	DAY		Real time
136	HOUR		
137	MINUTES		
138	SECONDS		
139	POINTER		Set by [40] SETPTR into macro or PA-422 buffer
140	RANDOM		Random # 0-255 (can be clipped)
141	HOURS	0-23	COM2: Time code
142	MINUTES	0-59	COM2:
143	SECONDS	0-59	COM2:
144	FRAMES	0-29	COM2:

If Parm 1 is in the range of 65-128 then the X variable (array+64) will receive the analog value from the analog channel specified in Parm 2. If Parm 2 is equal to 513 (which is the wildcard), then the channel number will be specified by the current X variable.

It is also possible to change the value of an "X" variable by receiving messages through the serial comm ports. The message consists of a 3 byte string, with a lead in character of \$FB (251 decimal), followed by the X variable to set in the range of \$01 to \$3F (or 1 to 63), followed by the binary value to set X() to, from \$00 to \$FF. Note that

only a value of 0 to 255 may be sent, even though internally X may be set to larger numbers when used for analog channels, timer values, etc.

Note that only the corresponding X variable in the array is changed, not the current value of the scalar "x" currently set by (and possibly in use by) one of the X variable event types. Use SetXY to actually use or test the value of a variable that has been received.

Use extreme caution when doing this, however, so that you don't accidentally change an X variable that is being controlled internally!

As of 10/16/97, this feature was added to COM2:.

Caution: The values 192-255 for P1 are not currently defined, but have been reserved for future expansion of X variables. DO NOT use these values for any X variable!!

- 21. X=X+p2** Increments variable X[PARM1] for counting applications. If PARM2=0 or 1 then X=X+1. If PARM2 > 1 then X=X+PARM2. If bit 7 (+128) of PARM1 is set, PARM2 is the number of another X variable that is added to PARM1. The scalar X variable is set, ready for testing.
- 22. X=X-p2** Decrements variable X[PARM1]. If PARM2=0 or 1 then X=X-1. If PARM2>1 then X=X-PARM2. Bit 7 set on PARM1 behaves as described above.
- 23. ModifyX** Modify the current 'X' variable based on the operation code in parameter 1, using the data (if appropriate) in parameter 2. The operations are listed in the table below:

P1 = Modify algorithm, P2 = DATA (if required)

PARM1	FUNCTION	DESCRIPTION
*0	(FUTURE)	
1	OR	Logically ORS value with
2	AND	Logically ANDS value with
3	XOR	Perform an EXCLUSIVE OR with
*4	SHL	Logical shift left 1 bit **
*5	SHR	Logical shift right 1 bit **
*6	ROL	Rotate left 1 bit **
*7	ROR	Rotate right 1 bit **
*8		
*9		
A	MIN	Forces a minimum value of (clipping)
B	MAX	Sets a maximum value of
*C	LOG	(Fixed translate to log table for audio)
*D	XLT	Translate from 256 byte table 2
*E	USR	Translate from table 3
F	RND	Sets X to a RANDOM number

\* Not implemented at this time \*\* may be changed to P2 # shifts

Note that a X=P2 using VARIABLE 129 can be used to resave an X variable after it has been modified.

- 24. If X =** **If X = (PARM1) ELSE SKIP (PARM2) EVENTS**  
This is a conditional test that may be used to process or skip a series of events based on a counter or toggle. "X" is the value of the last X modified or SET (26).

- 25. If X #**      **If X # (PARM1) ELSE SKIP (PARM2) EVENTS**  
Same as above, except the following events are executed if X <> PARM1, otherwise the number of (PARM2) events are skipped.
- 26. Set XY**      Forces the current X variable to X(PARM1) and Y to X(PARM2) without changing the value of either array variable. Up to 64 variables are available, numbered from 0 to 63. The current value of X is shown in the events window, i.e. X[32]=045. In addition, other system parameters and values may be tested, by specifying an X(PARM1) as follows. Many other event types allow wildcards from a SetXY to be used in place of PARM1 and/or PARM2.
- 27. if Run**      **IF RUN ELSE SKIP (PARM2) EVENTS - (PARM1 = 0)**  
Executes the following events if in normal run (playback) mode, will skip (PARM2) events if a fast-forward search would cause the events to be repeated or executed at the wrong time.
- 28. if HR=**      **IF HR = (PARM1) ELSE SKIP (PARM2) EVENTS**  
Allows testing the SMPTE hours field, and performing or skipping events as desired. An example would be for counting shows (knowing when to rewind), or for dynamically loading show data for different shows.
- 29. if HR#**      **IF HR# (PARM1) ELSE SKIP (PARM2) EVENTS**  
(Same as above with reverse logic.) NOTE: A skip value of 0 events is effectively a NOP.
- Whenever "X" or HRS are referenced, the current value will be updated on the right side of the EVENT window.
- All events used in a conditional group should have the same time set for execution, otherwise a REORG may mess things up!
- 30. Do\_Sub**      Performs an immediate call to an event Subroutine. Subroutines are built from the top of the events list and work down. They are entered in the format of Snnn.sss, where nnn is the subroutine number (1-250), and sss is a sequence number (used for jumping into the middle of a subroutine using Do\_Sub, separating delayed sections of a subroutine using the PAUSE command and for clarity and sorting purposes). Sequential events are executed as long as the subroutine number (nnn) equals the Do\_Sub number, thus there is no explicit "return" required. At this time, Do\_Subs may **NOT** be nested (i.e. called from within subroutines).
- Unlike ON:events, subroutines do not output data for each event executed. Once all events are scanned, the analog and digital channels are updated once for each frame. If absolutely required, a System (6) function can be used to force an update to all digital and analog output channels. Normally, ALL subroutines and events will be scanned, processed, and updated for each frame, and all outputs will be updated at the nominal frame rate.
- (96.12 and later) PARM2 can now be used to jump into a subroutine at a position other than the beginning and uses the "sorting" number (ie: S001.**XXX**) for the position to jump to within the subroutine.
- 31. ArmSub**      The subroutine (PARM1) will be executed if the trigger (PARM2) occurs while the system is executing.
- Triggers (PARM2) 1-10 represent the function keys 1-10. Triggers may also be received via the RS-232 COM ports 1 and 2. The format for software RS-232 triggers from COM1: or COM2: is a two byte message (\$FB header) followed by the trigger number.

Up to 64 subroutines may be armed at a time, and all armed conditions are cleared with a R.reset (hardware or software).

Currently, no error message is reported if an unarmed trigger, or for a trigger armed to a non-existing subroutine is received.

Only one subroutine may be armed to any given trigger at one time, and to disarm a trigger, either specify a subroutine (PARM1) with a 0 trigger, or a 0 subroutine for the trigger to disarm (PARM2).

- 32. ArmSec** Sets up a subroutine number (PARM1) that will execute after (PARM2) seconds of show time have elapsed. Note that the delay is relative to the actual show time code, and NOT the real time clock. It is legal to arm a delayed subroutine from within a subroutine. For more precise delays and timing for short term events, use function 33 (ArmFrm). The wildcard for PARM2 is 513 on CPU based systems and 255 for BART/LDC systems.
- 33. ArmFrm** Queues a subroutine number (PARM1) that will execute after (PARM2) frames of show time have elapsed. Note that the delay is based on the actual time code, and NOT the real time clock. It is legal to arm a delayed subroutine from within a subroutine. The wildcard for PARM2 is 513 on CPU based systems and 255 for BART/LDC systems.
- 34. FadeRt** Allows setting the ramp rate of analog channel (PARM1) to a multiplier of (PARM2). This allows automatic fades to be set up on any of the analog channels (lighting, audio, animation, etc.), as well as providing a damping factor for any analog values stored with a FadeTo. There is no wildcard for this function.
- *The first parameter is now a "dummy" argument. Once a fade rate has been set, multiple analog channels may be assigned to FadeTo new levels at that rate. Note that the rate is inversely related to time, i.e. a lower rate is a longer time. A rate of 32 corresponds to \_\_\_ ms.*
- 35. FadeTo** Sets the target level (PARM2) for a specified analog channel (PARM1), provided a FadeRt is also entered. The wildcard for PARM1 is 513 and PARM2 is 513 on CPU based systems.
- 36. MacOn** Engages the animation macro cue file (PARM1 = 1-32) for concurrent execution with other animation (and lighting and events) cues. Animation macros may terminate themselves after completion (indicated by an "x" on the macro display line, \$FF in the cue file), self-loop for chasing or "random" applications (\$FB in the cue file), or be killed with a MacOff (type 37 described below). An eight character free-form label may be assigned to each macro number, and will be displayed during any reference to the macro. Up to 32 animation macro files may be executing concurrently, either in normal or EMS memory. A value of 255 entered in PARM1 can be used as a wildcard. Note that a cue lock should be in effect for any .CUE animation data in normal memory.
- 37. MacOff** Terminates the macro (PARM1) started from above. A RESET operation disables all macro routines. A value of 0 terminates all running macros.
- 38. JamClk** Forces the system to jump to the minutes and seconds specified by PARM1 and PARM2. This is useful only when running in internal sync, as time code will force the system to update to the show time code point.
- 39. SetLok** When encountered as the FIRST event in a subroutine, checks to see if the lock channel (parameter #1) is already in use by another subroutine. If so, this subroutine is queued for later execution and aborted at this time. If the lock channel is clear, it is set to BUSY, and the subroutine is executed. After any (chain of) events is executed, the lock channel must be cleared by an Event type 40 (ClrLok).

The purpose of the locks are to enable multiple, asynchronous events and macros to share resources (such as serial ports, digital or analog channels, etc.) without conflict, and to allow enhanced interlocking for rigging or other show action equipment.

Lock numbers from 1 to 15 are allowed. Other numbers will be treated mod(16). Parameter 2 is the priority, from 1 (highest) to 15.

When a SetLck event is encountered, a scan is made to see if the lock associated with that channel is in use. If so, the subroutine number is added to a queue, and the execution of the rest of the subroutine is suspended, until another event (subroutine) unlocks the same channel number, at which point the next queued/suspended routine is restarted.

Normally, a SetLck would be the FIRST event in a locked subroutine, and an UnLock would be the last event of the subroutine or armed/delayed routine that has finished with the shared or locked resource.

#### Routine 1

S001. 001	SetLck	001	001	- Lock on channel 1 at priority 1
S001. 002	Di gOn	001	001	- Request a digital channel (i.e. mux)
S001. 003	QCom2:	000	001	- Send a message on RS-232 port
S001. 004	ArmDly	009	001	- Wait a second to send message

#### Routine 2

S002. 001	SetLck	001	005	- Lock on channel 1, at priority 5
S002. 002	Di gOn	001	002	- Request a digital channel (i.e. mux)
S002. 003	QCom2:	000	002	- Send a message on RS-232 port
S002. 004	ArmDly	009	001	- Wait a second to send message

#### Routine 3

S003. 001	SetLck	001	002	- Lock on channel 1, at priority 2
S003. 002	Di gOn	001	003	- Request a digital channel (i.e. mux)
S003. 003	Qcom2:	000	002	- Send a message on RS-232 port
S003. 004	ArmDly	009	001	- Wait a second to send message

#### Delayed routine to clear (and release lock)

S009. 001	Di gOff	001	001	- Release channel
S009. 002	Di gOff	001	002	- For both routines
S009. 003	Di gOff	001	003	- All three
S009. 004	UnLock	001	000	- Release lock for someone else

Now, if subroutine 1 is called, digital channel 1 is enabled, and a message is queued on the RS-232 port. If routine 2 is called, the system will detect that the lock is busy, and queue up a request to run routine 2 when the lock is cleared by the delayed routine. Further, if routine 3 is queued, while 2 is still pending, the priority is shifted so that routine 3 (priority 2) will execute next, followed by routine 2 (at priority 5). In all cases, a currently executing subroutine is allowed to complete, and will NOT be preempted by a higher level routine.

A R.eset from the Main menu will release all locks. Locks may be set or cleared in the regular events file, but have no inherent action until encountered when trying to start a subroutine.

#### **40. CtrLok**

Clears the locked channel specified by parameter #1. This should be performed when all resources or events have cleared the use of the locked resource(s). Locks may be used for serial ports, or to flag unsafe conditions until completion or reception of external signals, such as a lift or curtain reaching a safe position.

- 41. SetPtr** Sets a pointer within the pool of ASCII macro strings or the PA-422/general purpose buffer. This allows bytes to be read, modified, and rewritten (poked) for dynamic modification of the data within the buffer or macro pools. PARM1 specifies the macro (1-128) or PA-422 cue (1-64)+128 to access, PARM2 specifies the offset within the string or buffer pointer. The pointer is only set by this command. To read or peek into an "X" variable, use X(P1)=P2 (event type 20) with the pre-assigned variable function 139.
- 42. Poke** Only available in SCU Events.
- 43. Peek** Only available in SCU Events.
- 43. PA-422** Used to initiate a PA-422 cue at the time or subroutine specified. The first parameter is the PA-422 cue number. A PA-422 cue may either send or receive data from devices on the network, as specified within the PA-422 cue. Data within the send/receive buffer may be modified using the SetPtr, X= and other related event types.

*\*Note: PA-422 is only available in versions of Synthesis if actually needed.*

- 44. LDPcue** Used to initiate an LDP (Laser Disc Player) cue from within the events system, at the time (or trigger) indicated. The LDP cues carry their own start time information, and all accesses are automatically calculated and executed during random access/programming operations (without the event link). This event is used to link to and initiate LDP cues when playing in linear or normal playback mode. Parameter 1 specifies the cue number. The event time will normally be calculated as the cue start time, minus the number of frames specified in the pre-roll frame count.
- 45. LtCue** Invokes the lighting cue record number specified as the first parameter. Once initiated, the lighting cue and any linked lighting cues are on their own. PARM1 specifies the cue number and PARM2 may be used to override the fade time if <>0. Lighting cues are not valid in a TC-3500/3550 stand-alone controller. PARM1=255 uses X as cue and PARM2=255 uses Y as the fade rate.
- The cue number is the ABSOLUTE cue within the file, not the free-form eight character cue name used for script reference. The same cue may be used more than once in the event file; however, automatic lighting cue event entries are only created for the first (Time) entry as specified in the lighting cue.
- Caution - When lighting cues are inserted or deleted, an automatic update is made to the events file. Be sure to Save the EVENTS.EVT file in phase with the current LIGHT.LGT file whenever insert or delete changes are made to lighting cues!
- 46. --SB--** Subroutine identifier; allows entry of an eight character free-form label that will be displayed at the start of the subroutine (46). This also references the label to the subroutine for event types 30-33. This is an optional event type, and is used to aid in the readability of the events script. Subroutine names are not uploaded to LDC/BART Controllers.
- 47. Break\***
- 48. DigByt** Allows a bitmapped output of PARM2 to eight sequential digital channels, starting with the channel in PARM1. The channels align on 8 bit boundaries, beginning with channel 1, subchannel 1. This command may be used for parallel data output, and for matrix selection outputs such as a laser disc controller using a TC-388 Multiplexor card. Along with all binary values from 0 to 255, the following special values are possible:

256 - use the current Y variable as specified in the last SetXY

**49. CDRom**

Currently CD-ROM functions are handled by DC.COM driver. This driver is an interface between SYNTHESIS and MSCDEX (the DOS CD extensions driver) used to communicate to any generic CD drive capable of Red Book (CD Audio) playback. It allows audio to be cued and played internally and directly by Synthesis and to derive synchronization from the track time on the CD disc.

DC.COM must be loaded prior to starting Synthesis (normally invoked by a batch file). When DC is run, it attempts to read the VTOC (volume table of contents) and displays a list of tracks, starting times, and the length for each track. The resident portion will then reside in RAM. It is not necessary (or advisable) to reload the driver until the system is rebooted. If MSCDEX is not loaded, it will display an appropriate message and terminate.

This event sets up a software interrupt to the TSR driver, passing the following parameters as bytes in the AH/AL registers.

Parameters/operation for CD-ROM control are starting to emerge. PARM1 is used to specify the command or request to the CD-ROM drive(s).

PARM1	PARM2	DESCRIPTION
00*		Uninstall driver, release TSR
01*		Reinstall/initialize
02*	Baud	Set Baud. PARM2 is the baud rate for the comm port.
03	Drive #	Select drive. PARM2 of 0 is the first drive in the system, PARM2 of 1 is the second and so on (as opposed to using absolute drive ID's as previously documented).
04	0-99	SEEK track. The seek is assumed to be a TRACK, where the start/end locations are determined by the volume table of contents. DC.COM converts the track into the Red Book time code format and performs the seek to the start time of the track as determined when DC was started (or by a 49 19 03 (re-read VTOC) event).
05		PLAY - Plays the current CD track.
	0	The same start and length/end times as used in the seek will be used. Otherwise, it is handled the same as with a seek operation. Normally, a seek should be performed to pre-position the head in preparation for a synchronized start.
06		Seek Time - Used for LaserQ searching/cueing (internal). In future releases, it may be possible to seek to an absolute or relative time. In this case, PARM2 is 0-254, with the 16 character (extended) macro string used (and parsed) as the starting time and either LENGTH or END time, i.e. HHMMSSFF-HHMMSSFF (under 16 bytes).
07		Play Length - Determines end point for timed searches (internal).
08*		Set Time Code Mode
09*		Uninstall hooks/hardware interrupts. Reconnect with 01.
10*		Search to string (ASCII macro)
11*		Play TO string (ASCII macro)
12*		Jam Clock (Future)
13		Bad JuJu
14		Pause CD (after play)
15		Resume CD (after pause)
16		Return revision of driver
17		Return critical addresses (used internally)
18		Return time code (used internally by CD sync!)
	000	Read absolute time

	001	
	010	Read relative time (from start of track)
	011	
	100	
	101	
	110	
	111	
19		Send command
	01	Open Tray
	02	Close Tray
	03	Reread VTOC - Reloads the track information
	11*	SMPTE is absolute
	12*	SMPTE is relative, hours equal track number
20*		Return status (into "X") - Future

**NOTE: If PARM2 is specified as 255, the last Y value (from a SetXY) is passed as the parameter to the DC.COM driver.**

As an alternative to the 03 select drive command, the high hex nibble could be the unit code (if multiple drives or units are required), while the low nibble represents the command.

#### CD SYNC

In order to derive time code directly from the track time of the CD, it is possible to set the sync mode to "CD SYNC". This is currently possible using a SYSTEM (06) 006 000 event from within an events program, or by pressing "C" on the sync menu (even though it is not displayed as an option).

30 FPS code is derived by reading the current head position from the MSCDEX driver (through DC.COM) and converting the 75 frame time code to 30 FPS. This is then used as the reference time in place of SMPTE, pilot, or trace.

- **NOTE: We have had mixed results with this mode. On SOME CD drives/drivers, it works great. On other drives/drivers, the code returned is very erratic, with many missed frames. Try it with your drive and please report any results (success or failure) to us so that we can investigate. We suspect the problem is related to characteristics of certain drives or device drivers, but have not had enough units to identify the exact source of the problem.**

#### 50. Sound

This describes the interface between DOS/SYNTHESIS and various available multimedia functions. Parameter 1 is the function call, while parameter 2 is any related data based on the function used.

All interface and control is handled by a memory resident (TSR) program that must be loaded prior to starting Synthesis. The command line is "SB.COM /t". The /t is used to tell SB to terminate and remain resident. See the section in CPU EVENT labeled SB - Sound Sync Recorder/Editor for Sound Files for further information.

All Soundblaster functions are based on the 16 bit cards that incorporate revision 4 or higher of the DSP instruction set. Mixer functions take advantage of the extended resolution and features of the CT1745 or later mixer chips. The Soundblaster drivers are NOT used (in order to keep the code as compact, tight and fast as possible). In order to reduce the potential of conflicts, we recommend that you do NOT load the Creative Soundblaster drivers, and as such, it is NOT necessary to set any environment variables.

For VOICE (D/A) operations, pages 2 and 3 of the EMMS page frame are currently required. (Synthesis uses page 0 for MACRO EMS cues.)

The first Soundblaster card must be configured as follows:

IRQ = 5  
 DMA = 1 (8 bit DMA channel)  
 HMA = 5 (16 bit DMA channel)  
 BASE = 220h  
 MIDI = 330h

By fixing these values, the driver can be kept to a very small and efficient size, as no parsing or dynamic configuration is required.

### VOICE (SOUND FILE) FUNCTIONS

PARAM1	PARAM2	DESCRIPTION
4	mmm	OPEN - Where mmm equals ASCII macro of .TAF header file. Attempts to open the .TAF (Triad Audio File) header and read the parameters and file name for a .RAW audio file. The first three buffers (two DMA and a DTA area) are pre-loaded. This command will cause the time code to hesitate for a brief period as DOS takes control of the system. Please terminate the file name with a backslash "\" comment delimiter. Also, because "\" is used to delineate a comment, it may not be used to specify a path!
5	000	PLAY - Begins playback of the file. This call is made once to initiate playback of the first three buffers but will not refresh the buffers from the disk.
6	000	RFRSH - Called OFTEN (at least once every frame or two) to fill the DTA buffer from the disk and transfer data into the DMA buffer(s). If the file has not been opened or started in play mode, nothing bad will happen. So this process can be called continuously. For example,  <pre> 00: 10. 00 DoSub    011 000 ReFresh ..... S011. 000 --SB--   000 000 ReFresh S011. 001 Sound   006 000 S011. 002 ArmFrm  011 002 </pre> Future versions may not require this call to be made as we will attempt to "hook" into DOS in the background for all updates and buffering.
6	001	STOP - Stops playback and closes the header and audio files. This MUST be done to release the handles and buffer space back to the system if the file is not played to its physical end. I would also recommend putting one of these in the ON:RESET event list. Otherwise, the system may appear to hang or hiccup if a file was open and playing.
010	mmm	OPENWAV - Where mmm is a macro string pointing to a WAV file. This is a new command which allows wave sound files to be played directly, without the use of the .TAF header file. Note that currently we use the canonical WAVE file format, that is a RIFF file with a "fmt" chunk specifying the data format and one "data" chunk. Also, the SB driver will play out to the end of the file, so any bogus data beyond the end of the audio program will get played. (I've seen this is SOME .WAV files that have been downloaded from the Internet.) Valid in Revs 97.07 and later.

## MIXER FUNCTIONS

In the current implementation, all mixer functions are ganged such that the left and right channels track based on a single parameter for each source. These calls use the enhanced capabilities and resolution of the CT1745 or newer mixer chips, as used on the SB-16 and truly compatible cards.

For most of the level functions, the newer 5 bit level control is used giving a range of -62 to 0 DB in 2 DB steps. The level is normalized such that the three LSB's are ignored, anticipating even further/finer resolution in the level control.

PARAM1	PARAM2	DESCRIPTION
30	lvl	Where lvl is the 5 bit MASTER level (MSB's)
32	lvl	Where lvl is the 5 bit VOICE level (MSB's)
34	lvl	Where lvl is the 5 bit MIDI level (MSB's)
36	lvl	Where lvl is the 5 bit CD level (MSB's)
38	lvl	Where lvl is the 5 bit LINE level (MSB's)
40	lvl	Where lvl is the 5 bit MIC level (MSB's)
41	ggg	Where ggg = 000, 064, 128, 192 Resets the mixer to the default values and sets the output gain based on bits 6 and 7. The range is 18 DB in 6 DB steps.
43	00x	Where x is the MIC AGC control.
TBD	sss	Where sss is the LEFT switch control.
TBD	sss	Where sss is the RIGHT switch control.
		7      6      5      4      3      2      1      0
		MIDL MIDR LINL LINR CDL CDR MIC
	255	<b>NOTE: The current value of Y (from a SetXY) is used as the macro or level.</b>

## MIDI FUNCTIONS (\*\* NOT IMPLEMENTED\*\*)

The MPU-401 MIDI is used in UART mode and basically behaves like a standard serial port. The IRQ level and service is shared with the voice DSP processing, using IRQ 5 in all standard installations. The UART must be initialized (enter UART) prior to use and reset when the port is no longer required. The buffers are 256 bytes for input and output. At 30 FPS sampling, no more than 104 characters should be required per frame. Although interrupts are used to receive characters, no mechanism currently exists for interrupt driven output; rather, the data must be sent in a polled fashion. In order to facilitate this, the polling loop is available as an event call and will occur automatically during the VOICE output services.

PARAM1	PARAM2	DESCRIPTION
		Reset and enter UART mode
		Reset and clear UART mode
		Return pointer and # characters IN buffer
		Send pointer and # characters OUT buffer
		Perform polled output; PARAM2= max # characters

51. SndTrg Only in the OS9000 system.

52. Send X Only in the OS9000 system.

## CPU EVENT SYNTAX SUMMARY

	EVENT	PARM1	PARM2	COMMENTS
0	-----			Unused event
1	marker			Reserves a location in the events file
2	DigOn	(card)	(channel)	Turns on specified digital channel
3	DigOff	(card)	(channel)	Turns off specified digital channel
4	Pulse	(card)	(channel)	.5 second pulse to digital channel
5	AnValu	(channel)	(value)	Force analog channel to preset value
6	System	(function)		Operating system special functions
7	Pause	(seconds)	(frames)	Must be followed by new sort number
10	Com1:	(\$ or 0)	(macro)	Sends macro in PARM2 or ASCII in PARM1 to COM1:
11	Com2:	(\$ or 0)	(macro)	Sends macro in PARM2 or ASCII in PARM1 to COM2:
12	QCom1:	(\$ or 0)	(macro)	Queues/sends char or macro to COM1: port
13	Com3:	(\$ or 0)	(macro)	Sends macro in PARM2 or ASCII in PARM1 to COM3:
14	Com4:	(\$ or 0)	(macro)	Sends macro in PARM2 or ASCII in PARM1 to COM4:
15	XCom:	(port)	(macro)	Sends macro text (PARM2) to ports (PARM1)
16	Disply		(macro)	Display text of macro to screen
17	APatch	(logical)	(physical)	Redirects output of analog channel
18	AutoSt	(minutes)	(seconds)	Perform START macro after delay
19	Parall	(macro)		Sends the macro out the parallel port
20	Xp1=p2	(index)	(value)	Set variable/counter to a value or variable
21	X=X+p2	(index)		Bump X
22	X=X-p2	(index)		Reduce X
23	ModfyX	(function)	(data)	Modifies an X variable based on function
24	IF X =	(value)	(skip)	Skip steps unless X=constant
25	IF X #	(value)	(skip)	Skip steps if X <> constant
26	Set XY	(parm1)	(parm2)	Sets X=X[PARM1], Y=X[PARM2]
27	if RUN	(0)	(skip)	Skip cues if not in RUN mode
28	if HR =	(value)	(skip)	Skip steps unless hours=constant
29	if HR #	(value)	(skip)	Skip steps if hours <> constant
30	Do_Sub	(routine)	(sort #)	Routine 1-250
31	ArmSub	(routine)	(trigger)	As above when async event trigger occurs
32	ArmSec	(routine)	(seconds)	Activates event subroutine after delay
33	ArmFrm	(routine)	(frames)	Frame based delay
34	FadeRt	(channel)	(delay)	Sets a damping factor or fade rate
35	FadeTo	(channel)	(level)	Starts a ramp (FadeRt) for analog channel
36	MacOn	(1-16)		Begins an animation cue subroutine
37	MacOff	(1-16)		Forces a kill of a subroutine (1-16)
38	JamClk	(minutes)	(seconds)	Jams the internal clock to new time
39	SetLok	(lockchn)	(priority)	Sets a lock on lockchn
40	ClrLok	(lockchn)		Clears a lock in use on lockchn
41	SetPtr	(macro/cue)	(offset)	Sets a pointer to peek/poke buffer data
43*	PA-422	(cue number)		Execute the PA-422 cue specified
44	LDPcue	(cue number)		Perform the LaserView cue specified
45	LtCue	(cue #)		Execute lighting cue (internal)
46	--SB--			Identifies and labels a subroutine entry
47	Break			Used with an LDC/BART
48	DigByt	(channel)	(bitmap)	Set 8 digital channels to bitmap in PARM2
49	CDRom	(operation)	(parm2)	Interface with CD-ROM drive(s)
50	Sound	(operation)	(parm2)	Interface with Soundblaster or compatible audio card

## \* Not always available.

Events preceded by a "#" will NOT execute in "catch up" mode after a search. Events preceded by "?" will or will not execute based on the result of the last conditional test of "X" or HRS.

## ON: EVENTS

A special set of functions has been defined for the following special **events**, those at which the actual time is indeterminate or when time code may not be present. A subset of these functions is available on a TC-3500/3550 stand-alone show control or LDC (Laser Disc Controller) system.

- ON RESET** Defines what special actions are to occur when the computer is restarted, or the "R.eset" command is given from the computer keyboard, such as homing slides, resetting lights, effects, etc.
- Normally when the R.eset option is used, the clock is jammed to 00:00.00, the cue file and event cues are placed at their initial positions, and all channels are reset to off or a pre-defined startup value in the DEFAULT.DAT file.
- The "X" variables do not implicitly get reset to zero using R.eset or any other command. If desired, a **X=** event may be placed in the ON:RESET auto-event macro or subroutine.
- ON START** Defines the functions to execute if a remote operator START button is pressed, or the "S.tart" command is selected from the keyboard.
- Normally programmed to invoke a sequence to start a show, including an "S" (start deck) command to the TC-500 SMPTE Time Code Reader/Controller.
- ON ABORT** Performed if an external switch closure or "A.bort" is pressed. Can be programmed for an orderly shut-down in case of a break-down. Normally, this could be programmed to recue the system and allow a restart.
- Beginning with release 90.04, an ON:ABORT sequence will be performed whenever the ESC.ape key is pressed from the execute mode on a Synthesis host system.**
- ON ERROR** Defines a sequence of events if an external error condition (i.e., a film break) occurs. Normally, manual operator intervention would be required to clear the error condition. Any emergency stop (E-STOP) controls should be wired to this sequence.
- ON PAUSE** Defines operations to be performed if a pause button is pressed, or a restart sequence if used as a continue from an abort or error.
- START, ABORT, ERROR, and PAUSE** are normally invoked from external switch closures. Refer to the TC-500 SMPTE manual.
- ON END** Requests the actions to be taken if an "END ENABLE" event has been entered into the EVENT list **and** the SMPTE code is lost for more than one second (at the end of the show). Used to define the sequence at the end of a show, but not necessarily the physical end of the reel, cassette, or loop.
- ON EOT** Analogous to "ON END", except this sequence marks the physical end of the tape (or last show on a reel), and can be used to issue a rewind or recue to the beginning of the film or tape. One way to differentiate the last show when running under SMPTE is to include enough extra code in the last show to cause the ON:EOT event to be processed, overriding an END.
- ON END and ON EOT are incorporated as a fail-safe against momentary (deliberate or accidental) loss of time code, stopping the tape, or whatever. Only when an ON END or ON EOT event is enabled and expects the actual end of the show and/or reel of tape is the appropriate event sequence invoked.

**ON INIT**

Special sequence of up to 16 steps that is performed only ONCE, when the system is first started. External devices requiring a special startup procedure may be initialized using this feature. Examples include spinning up laser discs, homing projectors, turning on motors, non-dims, etc.

These special **events** are located above the normal stack of 1000 time-based event cues, and are referenced when the specific condition occurs.

It is possible to simulate an external "ON:" event trigger (for debugging) by entering the number corresponding to the on:event type from the EVENT menu. (Use the upper row of numeric keys.)

## EVENT ENTRY AND EDITING

From the Events menu, select "E.dit (events)". Sixteen events are displayed, although all entry and editing occurs on the center line. For editing operations, the NumLock key should be OFF, so that the arrow and function keys will be interpreted correctly. Use the UP (8) or DOWN (2) arrow keys to scroll forward or backwards one event at a time. The "PgUp" and "PgDn" keys may be used to advance a full page (16 events) at a time.

The cursor will flash at the start of the time field. To accept the time displayed and move to the type field, just press <return>; otherwise enter the correct time using the format MM:SS.FF (use the -> forward arrow key to copy over any numbers you wish to keep).

An offset may be made to the current time by preceding the time with a "+" (to add MM:SS:FF to the current time) or "-" (to subtract). Separate each number with a comma or space. If only one number is entered, it is assumed to be a frame offset; two numbers mean seconds and frames, and three numbers may be used to offset MM, SS, and FF. Once an offset is entered, it may be repeated on subsequent lines using the F3 function key.

To go directly to the special set of ON:events, use "O.n:" instead of "E.dit". This is also convenient to find the end of the event subroutines, which are built just below the ON:events and work down in the file. To find and edit the events subroutines, use the "S.ubr" command.

Note that the "time" field of one of the special "ON EVENT"s can not be changed. Don't even try!

Pressing the **END** key ("1" on the numeric pad) will cause the contents of the current event to be zapped, available for entry of a new event. The spaces left after deleting events may be cleaned up by using the R.eorganize, or by entering new events.

As of release 7.4, F10 will also perform the delete function, and F9 will insert an empty cue at the current position, pushing all other events up to an empty marker up by one. Note that in the event subroutine area, it may not be possible to insert, as doing so would cause the last event to be lost off the top (past 999).

The next field is the **TYPE** of event. A menu of today's choices will automatically appear in the right side of the events window. Enter the option number from the list (if changing), followed by the <return> key. The Up and Down arrow keys may also be used to select an event type from the menu bar.

To view additional menus of event types, use the "PgUp" or "PgDn" key on the numeric pad.

Normally, all events are executed, in order, up to the current time code location. In many cases, and especially during programming, it is not necessary to execute certain cues (i.e. strobe flashes) unless the show is being run end to end. Placing a "#" (pound) sign in front of the event type number will set a flag which causes the event to be skipped during searches, and executed only if the time code matches the event time.

The third and fourth fields are parameters used depending upon the particular type of event (card/channel numbers, analog value, macro selection, etc.). Normally, the numbers are entered as decimal values in the range of 0 to 255; however, if you prefer hexadecimal, precede the number with a "\$" (e.g. \$FF = 255). Finally, if the parameter calls for a single ASCII character (such as an "S" start command to the TC-500 SMPTE controller), enter a "\$" (hex) "" (quote) and the character (remember to use the shift key if you want an upper case letter). When you press <enter>, the character will be converted and displayed as a **decimal** value.

Use the ESC.ape key to abort all changes to a line if you are **not** in the TIME field. Pressing ESC.ape while in the time zone will return you to the EVENTS menu.

## LOADING AND SAVING EVENTS

At startup time, a default file name of "EVENTS.EVT" is used to load the initial events and macro data unless overridden in FILES.DAT. When making changes to any of the events or macros, after Reorganizing the file, be sure to Save the file to disk. If saving to a floppy disk, be sure the diskette is in the drive, unprotected, and that the drive door is closed. You will be prompted for the name of the file to save. DOS filename conventions must be followed, using up to eight letters, a "." and an extension. We recommend an extension of ".EVT" or some meaningful name that will uniquely identify the type of data the file contains. Files are loaded or saved in the current directory of the selected drive.

**L**.oad and **S**.ave operations require that the <shift> key be used.

If the current name is acceptable, just press <enter>; otherwise, type in the path, file name and <enter>. Similarly, a new events file can be Loaded, **replacing** the contents of memory. Use the ESCape key to cancel a Load or Save request. When saving a file, a prompt will be made to "Do Backup?". Pressing "Y" or <enter> indicates that a backup should be made, pressing "N" or <escape> indicates that a backup file will not be made, and the current events data will replace any existing file. A backup events file has the extension of ".\$VT". When working on a hard disk, be sure to copy (all) working files to a floppy disk as additional protection to your work. Note that saving events does not affect any of the other files that are used by the system, including LIGHTING cues, ASSIGNMENTS, or names files.

## REORGANIZING EVENTS

After editing the time or subroutine fields, Merging data from another file, or sorting MARKERS (placed during programming) into the file, use the Reorg(anize) function to resequence the events in ascending order by time and by subroutine number and sequence. This is necessary so that all events will be processed at the correct time and sequence, and that subroutines can be properly located. An event **and all** subsequent events **will not be processed until the show time is equal to or greater than the next** queued event! Entering or editing lighting cues will automatically reorganize the EVENTS file.

Subroutines will be found just below the ON:RESET macro, and will be sorted according to subroutine number; none of the ON:event sequences are affected by a Reorg.

The Events window on the main screen will always display the time, type, and parameters of the **last** event to be processed (top line) and the **next** event to occur (second line).

## MERGING EVENT FILES

**CURRENTLY, MERGE IS NOT IMPLEMENTED. Contact Triad for a utility for merging events.**

## MARKING EVENTS

It is possible to Mark show code times in the events file while executing the show under time code synchronization. While playing the show tape (in Execute mode), press the "M" key for every event you wish to flag. Once the event time(s) are marked, use the Events editor to change the MARKER (1) events to the type of event (with associated parameters) desired. Marked events are stored above current regular event cues in memory, and may easily be edited or Reorganized (sorted) into the active sequence. Don't forget to Save all newly entered events to disk! Events types and parameters may be entered at the end of the timed events list, but with a null time value of "--:--:--". Then, when the mark key is pressed, the event will execute and the time will be recorded into the event during a real-time performance.

## MACRO ENTRY AND EDITING

Up to 128 32-byte ASCII character strings may be entered as macros to be sent to either the COM1: serial port normally used for a TC-500/TC-750S SMPTE controller), the COM2: serial port, one of the parallel ports, or the screen. ASCII string macros are often used to send messages to video disc players or other serial controlled devices, but may also be used to specify file names or messages to prompt a user and provide status.

Press "M" (M.acro) to edit the message strings, then using the cursor keys, select the line you wish to edit, and key in the text. (Refer to the document on the INPUT ROUTINE for advanced editing operations and shortcuts.)

The "**PgUp**" and "**PgDn**" keys are used to select up to eight pages of 16 macro definitions.

In order to shorten the string (less than 32 characters) and to allow comments in the macro field, the "\" (backslash) character has been reserved as a delimiter, which ends transmission of text from the "\" to the end of the line. As the "ESC.ape" key would normally be used to exit the editor, it is possible to transmit an ASCII escape sequence (27d, 1Bh) by substituting the "|" (shift-backslash) character.

Any ASCII character may now be sent by placing the decimal value of the character after a "/" (forward slash). The next non-numeric character is used as a break character, and the macro continues with the following exceptions:

/nnX	send the ASCII character represented by X variable = X(nn)
/nnx	send an ASCII string representing the number of the X variable
/nnA	send the ASCII character represented by analog channel nn
/C	send a checksum byte = to all of the characters sent in this macro
/c	send an inverted checksum (256-/C)

Reserved characters in the string perform special functions including:

\	(the rest of the line is a comment)
/	(start building a number as described above)
%	reset the current running checksum (to 0)
&	(as the last character on a line) - continue the macro on the next line

Other special ASCII codes may be sent using the direct form of the event (that is, where PARM2 is 0) if they cannot be entered from the keyboard. An example is a CR or LF character.

The MACRO definitions are saved and loaded automatically with the events file data. Therefore be sure to S.ave the EVENTS data if changes are made to macros! Event types that refer to ASCII macros will display the first eight characters of the macro as a reference.

## SONY LDP CODE CHART AND APPLICATION

Please refer to the reference section that contains the common commands used by the SONY series of Laser Disc Players, including the LDP-1200, LDP-1250, LDP-1500, and LDP-2000. These commands will work on a basic level I machine.

### Application:

We recommend that a "V" clear buffer command be placed in front of every search string to ensure that the player's command buffer is flushed from garbage characters or partial command strings. The Sony machines will display "INVALID COMMAND" on the video output if a bad command is sent, which is to be avoided. The Sony LDP series will operate at 1200 - 9600 baud, but do not stack or buffer commands. We have had occasional difficulties operating at 9600 baud; if problems are encountered, try a slower baud rate, or try using the QCOM events.

The following basic sequence may be used to cue to a starting frame number, then play to a target frame number. A minimum of 2.5 seconds should occur between the search and play commands.

VC01000@\	clear buffer, then search to frame 1000, @=enter
VD02000@1@\	clear buffer, play to frame 2000, enter, play once, enter

Normally the comment field would contain a description of the segment being played. If the disc is being setup to play "forever", use a 0 for the repeat count (the 1 above tells the player to play the segment once). If the second frame number is less than the first, the disc will play normal speed, backwards albeit without audio.

## PIONEER 8000 CODE CHART

Please refer to the reference section that contains the common commands used by the Pioneer LD-V8000 Laser Disc Players.

**SB - SOUND SYNC RECORDER/EDITOR FOR SOUND FILES**

Rev 1.2 07/28/97 wjs

This document describes the operation of the SB program in interactive mode, used to record, play, and prepare the necessary files for use under Synthesis.

In our application, all sound files themselves are "raw" data, without any of the embedded codes and headers as found in .WAV or .VOC files. Rather, we use a separate header file (with the extension .TAF) that specifies the filename for the .RAW sound file, along with parameters such as sampling rate, number of bits/sample, stereo/mono, etc. and timing information reserved for future applications.

As there are literally dozens of commercially available applications for complex editing and manipulation of audio files, this program has not been fully developed. At this time, SB supports only the bare essentials of basic recording, creating the header file, and its primary function of being a TSR support for playback of files under Synthesis show control. For the majority of sound files, I would recommend using the .WAV file format.

Likewise, as there are several TSR programs for controlling the mixer, a user interface to the mixer is not included in SB. Use SB16MIX (or an equivalent provided by Creative Labs) as a pop-up for mixer control. When used as a TSR, mixer control can be achieved with the system calls as described in the CPU EVENT document, system commands 49 (CDRom) and 50 (Sound).

SB is currently hardcoded to use an official Creative Labs Soundblaster 16 audio card, with revision 4.0 or later firmware. All I/O is done directly with card at the hardware level, so look-alikes may NOT work. As a result, NONE of the Creative Labs low-level drivers are required (or even desired) in your AUTOEXEC.BAT or CONFIG.SYS files.

When using SB, bear in mind that disk usage is extensive, especially at the higher sampling rates and in 16 bit/stereo mode. Thus, it is essential that you use a fast disk drive and preferably one designed for multi-media use.

To start the program, type SB from the DOS prompt. Without the /t option, the program will come up in interactive mode, with simple menu driven command options.

The first menu is the "top" level. There are several options that have sub-menus in a tree structure. Press the letter corresponding to the desired function to initiate options. The arrow keys and mouse are not supported at this time (again, this is a very primitive preliminary program!).

TOP: F.ile L.evels P.lay R.ecord S.ettings T.ime Q.uit/reside eX.it

File: xxxxx N.ame -> R.aw xxx xxx U.pdate ESC

N.ame Refers to the header file, as described above. We normally use .TAF as a way to identify sound header files, although any valid nnnnnnnn.xxx DOS file name may be entered here. If the header already exists, you can proceed to P.lay the file. (This must be done to actually open the file if you intend to modify existing S.ettings!)

R.aw Refers to the actual raw sound binary image, either to be recorded on this system or imported from some external source.

U.pdate Forces the header information (the name of the .RAW file and the sound playback parameters) to be written to the .TAF header file. Note that this is a MANUAL operation that must be performed if the R.aw name or any of the settings are changed.

Other possible future options are shown on the file menu. It is anticipated that we will offer conversion to and from .WAV and .VOC file formats and other little goodies. Someday. Maybe.

**L.levels** Used to set recording levels. It operates exactly like R.ecord except that the file is not actually created.

At the present time, levels are displayed in HEX towards the end of lines two (left channel) and three (right channel). The numbers to the left are the peak values encountered in the last block of data, while the numbers to the right are the peak values for the entire sample. The idea is to sample the entire audio segment, and ensure that the level never clips (FFFF in 16 bit mode), yet gets as close to the peak value as possible at the loudest moment. Eventually, a fancier bar graph display may be implemented.

**P.lay** Attempts to open the header file as specified by the F.ile N.ame. If successful, the header is read, including the name of the associated .RAW file, the playback parameters, and time information described below. A "POP?" message appears on the bottom of the screen, and the program awaits a key press of the space bar, or receipt of the matching SMPTE time code from a Triad SMPTE Reader (such as a TC-750 or TC-550 BART). The file is played back with the current running time and levels displayed in real time. At this time, the system will not "chase" SMPTE; it only waits for a matching start time to begin playback, then it's on its own!

**R.ecord** Used to sample audio from the Soundblaster's line inputs and record the data into a .RAW file using the current S.ettings for sampling rate, mono/stereo, etc. It is recommended that you first use the L.levels option and play the audio all the way through to ensure that the optimum recording will be made.

The Soundblaster mixer (especially the mic preamp) is very noisy. We recommend that you use the mixer controls to fully reduce and mute all unused inputs when recording sound samples!

During recording you can use the S and T keys to mark the secondary and tertiary cue points (time references) reserved for future applications, i.e. sending a trigger to an external device or program.

**S.ettings** A submenu of the various parameters for the associated .RAW sound file. The various settings are logged in the .TAF header file using the F.ile U.pdate option and determine how the file will be recorded or played back. There are obvious trade-offs in how the settings are used, on a file-by-file basis. The following chart shows the relationships of space versus the various parameters.

<b>DIGITAL AUDIO SYSTEM BYTE/TIME CODE CALCULATIONS SPACE REQUIREMENTS</b>				
<b>SAMPLE RATE</b>	<b>8 BIT MONO</b>	<b>8 BIT STEREO</b>	<b>16 BIT MONO</b>	<b>16 BIT STEREO</b>
<b>BYTES/MINUTE</b>				
11.025	661500	1323000	1323000	2646000
22.05	1323000	2646000	2646000	5292000
33.075	1984500	3969000	3969000	7938000
44.1	2646000	5292000	5292000	10584000
<b>BYTES/SECOND</b>				
11.025	11025	22050	22050	44100
22.05	22050	44100	44100	88200
33.075	33075	66150	66150	132300
44.1	44100	88200	88200	176400
<b>BYTES/FRAME (30FPS)</b>				
11.025	367.5	735	735	1470
22.05	735	1470	1470	2940
33.075	1102.5	2205	2205	4410
44.1	1470	2940	2940	5880
<b>FRAMES/BUFFER (16384)</b>				
11.025	44.582313	22.291156	22.291156	11.1455782
22.05	22.291156	11.145578	11.145578	5.57278912
33.075	14.860771	7.4303855	7.4303855	3.71519274
44.1	11.145578	5.5727891	5.5727891	2.78639456
<b>SECONDS/MEGABYTE</b>				
11.025	95.108934	47.554467	47.554467	23.7772336
22.05	47.554467	23.777234	23.777234	11.8886168
33.075	31.702978	15.851489	15.851489	7.92574452
44.1	23.777234	11.888617	11.888617	5.94430839

The various options are self-explanatory, so let me explain them:

- B.its Toggles between 8 bit and 16 bit samples
- M.ono Selects single channel record/play mode
- S.tereo Selects dual channel mode
- \*R.ate (Future - Will allow a non-standard rate to be selected)
- 5. Selects a sampling rate of 5000 samples/second
- 1.1 Selects a sampling rate of 11.025 Kb/s
- 2.2 Selects a sampling rate of 22.050 Kb/s
- 3.3 Selects a sampling rate of 33.075 Kb/s
- 4.4 Selects a standard sampling rate of 44.1 Kb/s
- \*C.onfig (Future - Allows selection of different sound cards, etc.)
- \*P.op (Future - Allows the selection of record start options, i.e. by time code, key, contact closure, or first audio threshold)
- ESC Ubiquitous back up key

Beginning with 97.03, a "dirty" flag is set when some of these options are used, such that you will be prompted to update the header file settings.

- T.ime      Time is used to set/edit some of the time fields related to the audio file and stored in the .TAF header file. For instance, a SMPTE start time may be specified that is used to start the file based on an absolute SMPTE time code received from the comm port. These fields are reserved for future applications that may require them and are not necessary for any of the Synthesis functions when acting as a TSR driver.
- Q.uit/reside      Exits the SB program, leaving it resident for use by Synthesis (or other programs) to control the mixer and allow playback of audio files in the "background". When the program is "Q.uit", all of the code used for the interactive user interface are returned to DOS, and only the stub code required for TSR operation remain in memory. It is not smart enough to realize this if you attempt to run the program again, so eventually your memory will get gobbled up (plus the operation may become erratic or unstable). Thus, please re-boot before re-running the application. For normal playback use, it is best to just start the program with the /t (TSR) switch.
- eX.it      Exits the SB program completely, removing all resident parts.

We believe this information to be correct as of publication, but assume no liability for damages or injury (consequential or inconsequential) incurred by the use or application of this software. Please report any errors or omissions in this documentation or in the program. All specifications and operational procedures subject to change.

© 1984-1999 William J. Synhorst/All rights reserved. This information and the associated software and algorithms is proprietary and confidential and represents trade secrets and original work of the author, and may not be reproduced, revealed, or disclosed without written permission of William J. Synhorst or Triad Productions, Inc.