

Appendix A. Multifunction Show Controller Showgram Definitions

Words in italics are non-terminal symbols, i.e., they are not final; instead, they are made up of smaller things: tokens or other non-terminal symbols. Non-italics words are tokens, or keywords. Keywords are case insensitive. Variables defined outside of any sequences are considered global and can be referenced inside any sequence. However, if a local variable has the same name of a global variable, the local variable will be used inside the sequence in which the local variable is declared. Some commands do not generate any pseudo code, e.g. relative, autorun and autostart. They can be placed anywhere inside the sequence, even inside a loop. Some expressions require specific type of variables. For example, the bitwise or '|' is applicable only to bit variables, opto, tally, relay, skey and logical constants (TRUE, FALSE, ON, OFF, OPEN, CLOSED etc). The compiler will generate error messages if the operator and the operands do not have compatible types.

program:

```
    sequencelist  
|    declarationlist sequencelist
```

declarationlist:

```
    declaration  
/    declarationlist declaration
```

declaration:

```
    bit variable ;  
|    byte variable ;  
|    time variable ;  
|    long variable ;          /* time and long are identical types */  
|    string variable ;       /* strings are constant strings */
```

sequencelist:

```
    sequence  
/    sequencelist sequence
```

sequence:

```
    sequence variable { statementlist }  
/    sequence variable : timestring , timestring { statementlist }
```

statementlist:

```
statement
/  
statementlist statement
```

statement:

```
command ;  
/  
timestring command ;  
/  
label : command ;  
/  
label : timestring command ;  
/  
declaration
```

command:

```
/  
variable = expression /* can be long, time, byte or bit */  
/  
variable += expression /* can be long, time or byte */  
/  
variable -= expression /* can be long, time or byte */  
/  
variable *= expression /* can be long, time or byte */  
/  
variable /= expression /* can be long, time or byte */  
/  
variable %= expression /* can be long, time or byte */  
/  
variable |= expression /* can be byte or bit , comparisons are bit expressions */  
/  
variable &= expression /* can be byte or bit , comparisons are bit expressions */  
/  
variable ^= expression /* can be byte or bit , comparisons are bit expressions */  
/  
variable = sexpression  
/  
tally = expression /* must be bit expression */  
/  
relay = expression /* must be bit expression */  
|  
extvar = expression /* can be byte or long expressions */  
|  
nop  
|  
autorun /* sequence will loop forever from first to last statements*/  
|  
autostart /* sequence will run at power up or after download */  
|  
relative /* time stamps on the left of the statements are relative to sequence entry time.  
When “forever” is defined, entry time is reset everytime a wrap around happens.  
If “relative” is not defined, then time stamps are absolute real time. */  
|  
clear variable up /* clear the up transition flag */  
|  
clear variable down /* clear the down transition flag /  
|  
clear relay up /* clear the up transition flag */  
|  
clear relay down /* clear the down transition flag */  
|  
clear tally up /* clear the up transition flag */
```

```

| clear tally down /* clear the down transition flag */
| clear opto up /* clear the up transition flag */
| clear opto down /* clear the down transition flag */
| clear skey up /* clear the up transition flag */
| clear skey down /* clear the down transition flag */
| display sexpression to line /* display a string expression to the given line */
| display sexpression to line at number /* display a string expression to the given line at the given cursor position */
/ display expression to line /* display the result of the expression to the given line */
/ display expression to line at number /* display the result of the expression to the given line
at the given cursor position */

/ display port to line /* display the receive buffer of the given port, receive buffer untouched */
/ display port to line at number /* display the receive buffer of the given port, receive buffer untouched */
/ display number from port to line /* display the given number of characters from the receive buffer of the given
port . Receive buffer untouched. */

/ display number from port to line at number /* display the given number of characters from the receive buffer of the given
port . Receive buffer untouched. */

| jump label /* jump to the line with the given label */
/ if expression then jump label /* if expression evaluated to true, then jump to label */
/ if not expression then jump label /* if expression is false, then jump to label */
/ if variable high in variable then jump label /* if variable goes high in the given time then jump to label */
/ if variable low in variable then jump label /* if variable goes low in the given time then jump to label */
/ if variable up in variable then jump label /* if variable goes up in the given time then jump to label */
/ if variable down in variable then jump label /* if variable goes down in the given time then jump to label */
/ if relay on in variable then jump label /* if relay energized in the given time then jump to label */
/ if relay off in variable then jump label /* if relay de-energized in the given time then jump to label */
/ if relay up in variable then jump label /* if relay goes up in the given time then jump to label, up means a transition from
de-energized to energized */

/ if relay down in variable then jump label /* if relay goes down in the given time then jump to label, down means a
transition from energized to de-energized */

/ if tally on in variable then jump label /* if tally turned on in the given time then jump to label */
/ if tally off in variable then jump label /* if tally turned off in the given time then jump to label */
/ if tally up in variable then jump label /* if tally goes up in the given time then jump to label, up means a transition from
off to on */

/ if tally down in variable then jump label /* if tally goes down in the given time then jump to label, down means a transition
from on to off */

```

/	if <i>opto</i> closed in <i>variable</i> then jump <i>label</i>	/* if <i>opto</i> closed in the given time then jump to label */
/	if <i>opto</i> open in <i>variable</i> then jump <i>label</i>	/* if <i>opto</i> open in the given time then jump to label */
/	if <i>opto</i> up in <i>variable</i> then jump <i>label</i>	/* if <i>opto</i> goes up in the given time then jump to label, up means a transition from closed to open */
/	if <i>opto</i> down in <i>variable</i> then jump <i>label</i>	/* if <i>opto</i> goes up in the given time then jump to label, up means a transition from closed to open */
/	if <i>skey</i> closed in <i>variable</i> then jump <i>label</i>	/* if a scanned key closed in the given time then jump to label */
/	if <i>skey</i> open in <i>variable</i> then jump <i>label</i>	/* if a scanned key open in the given time then jump to label */
/	if <i>skey</i> up in <i>variable</i> then jump <i>label</i>	/* if a scanned key goes up in the given time then jump to label, up means a transition from closed to open */
/	if <i>skey</i> down in <i>variable</i> then jump <i>label</i>	/* if a scanned key goes down in the given time then jump to label, down means a transition from open to closed */
/	wait for <i>variable</i> high then jump <i>label</i>	/* wait for bit variable to go high then jump to label, a blocking call */
/	wait for <i>variable</i> low then jump <i>label</i>	/* wait for bit variable to go low then jump to label, a blocking call */
/	wait for <i>variable</i> up then jump <i>label</i>	/* wait for bit variable to go up then jump to label, a blocking call. Up means a transition from low to high */
/	wait for <i>variable</i> down then jump <i>label</i>	/* wait for bit variable to go up then jump to label, a blocking call. Up means a transition from low to high */
/	wait for <i>tally</i> on then jump <i>label</i>	
/	wait for <i>tally</i> off then jump <i>label</i>	
/	wait for <i>tally</i> up then jump <i>variable</i>	
/	wait for <i>tally</i> down then jump <i>label</i>	
/	wait for <i>relay</i> on then jump <i>label</i>	
/	wait for <i>relay</i> off then jump <i>label</i>	
/	wait for <i>relay</i> up then jump <i>variable</i>	
/	wait for <i>relay</i> down then jump <i>label</i>	
/	wait for <i>opto</i> closed then jump <i>label</i>	
/	wait for <i>opto</i> open then jump <i>label</i>	
/	wait for <i>opto</i> up then jump <i>label</i>	
/	wait for <i>opto</i> down then jump <i>label</i>	
/	wait for <i>skey</i> closed then jump <i>label</i>	
/	wait for <i>skey</i> open then jump <i>label</i>	
/	wait for <i>skey</i> up then jump <i>label</i>	
/	wait for <i>skey</i> down then jump <i>label</i>	

/	wait for <i>variable</i> high	/* wait for bit variable to go high then continue, a blocking call */
/	wait for <i>variable</i> low	/* wait for bit variable to go low then continue, a blocking call */
/	wait for <i>variable</i> up	/* wait for bit variable to go up then continue, a blocking call. Up means a transition from low to high */
/	wait for <i>variable</i> down	/* wait for bit variable to go up then continue, a blocking call. Up means a transition from low to high */
	wait for <i>tally</i> on	
	wait for <i>tally</i> off	
	wait for <i>tally</i> up	
	wait for <i>tally</i> down	
	wait for <i>relay</i> on	
	wait for <i>relay</i> off	
	wait for <i>relay</i> up	
	wait for <i>relay</i> down	
	wait for <i>opto</i> closed	
	wait for <i>opto</i> open	
	wait for <i>opto</i> up	
	wait for <i>opto</i> down	
	wait for <i>skey</i> closed	
	wait for <i>skey</i> open	
	wait for <i>skey</i> up	
	wait for <i>skey</i> down	
/	if <i>port</i> error then jump <i>label</i>	/* if given port failed to transmit or receive then jump to label */
/	<i>tally</i> type = <i>tallytype</i>	/* set the given tally to the given type, either PWM or SIMPLE */
/	<i>tally</i> dutycycle = <i>number</i>	/* set the given tally's PWM to the given duty cycle, if not in PWM mode, duty cycle will still be set and remembered. */
/	pwmfreq = <i>number</i>	/* set the PWM frequency, affects all 8 tallys, 1 to 8 */
	pause <i>variable</i>	/* pause a given sequence */
	start <i>variable</i>	/* start a given sequence */
	stop <i>variable</i>	/* stop a given sequence */
	resume <i>variable</i>	/* resume a given sequence */
	idle until <i>expression</i>	/* wait until the system clock equal to the given expression (in frames or time) */
	idle until nextday	/* wait until the next day */
	idle for <i>expression</i>	/* wait for a period (in frames or time) defined by the given expression */

	keyscan <i>number</i> , <i>number</i>	/* start key scanning. The first number defines number of rows (tally outputs) and the second defines the number of columns (opto inputs) */
	keyscan high	/* set tally high for key scanning */
	keyscan low	/* set tally low for key scanning */
	framesync	/* wait till the beginning of next frame /
	reset reference	/* reset the current time to zero when the “relative” flag is set */
	reference = <i>expression</i>	/* set the current time to the result of <i>expression</i> when the “relative” flag is set */
	<i>port</i> type = <i>devicetype</i>	/* defines the serial port type, LD, ASCII, DMX or MIDI */
	reset <i>port</i>	/* reset the given serial port */
	hold <i>portlist</i>	/* hold the given list of ports, comma separated */
	release <i>portlist</i>	/* release the given list of ports, comma separated */
	send <i>variable</i> to <i>port</i>	/* send the content of the given string variable to the given port */
	send <i>sexpression</i> to <i>port</i>	/* send the result of the given string expression to the given port */
	send <i>variable</i> to host	/* send the content of the given string variable to the host port */
	send <i>sexpression</i> to host	/* send the result of the given string expression to the host port */
	<i>port</i> baudrate = <i>number</i>	/* set the given port’s baudrate, number must be 300, 1200, 2400, 4800, 9600, 19200, 31250, 38400, 57600 or 115200 */
	<i>port</i> parity = <i>parity</i>	/* set the port parity, parity must be none, odd, even, high or low */
	<i>port</i> wsize = <i>wordsize</i>	/* set the port character size, 5bits, 6bits, 7bits or 8bits */
	<i>port</i> stopbit = <i>stopbits</i>	/* set the port stopbits, one, one&half or two */
/	empty <i>number</i> from <i>port</i>	/* remove the given number of characters from the receive buffer of the given port */
/	empty <i>port</i>	/* empty the receive buffer of the given port */
	empty host messages	/* empty host message buffer */
	ldsearch <i>port</i> to <i>expression</i>	/* search the laser disk at the given port to the frame given by the result of the expression */
	wait for ldsearch <i>port</i> to <i>expression</i>	/* as above but wait for the laser disk to return ‘R <CR>’ */
/	ldsearch <i>port</i> to <i>sexpression</i>	/* search the laser disk at the given port to the location defined by the ascii expression. This is useful for CLV disks or CD */
	wait for ldsearch <i>port</i> to <i>sexpression</i>	/* as above but wait for the laser disk to return ‘R <CR>’ */
	ldplay <i>port</i>	/* start playing the laser disk at the given serial port */
	wait for ldplay <i>port</i>	/* as above but wait for the laser disk to return ‘R <CR>’ */
	ldplay <i>port</i> to <i>expression</i>	/* play the laser disk at the given port to the frame number defined by the expression */
	wait for ldplay <i>port</i> to <i>expression</i>	/* as above but wait for the laser disk to return ‘R <CR>’ */

	<i>ldplay port to sexpression</i>	/* play the laser disk at the given port to the location defined by the ascii expression */
	<i>wait for ldplay port to sexpression</i>	/* as above but wait for the laser disk to return 'R <CR>' */
	<i>ldstop port</i>	/* stop the laser disk at the given serial port */
	<i>wait for ldstop port</i>	/* as above but wait for the laser disk to return 'R <CR>' */
	<i>ldstart port</i>	/* spin up the laser disk at the given serial port */
	<i>wait for ldstart port</i>	/* as above but wait for the laser disk to return 'R <CR>' */
	<i>ldstill port</i>	/* freeze the laser disk at the given serial port */
	<i>wait for ldstill port</i>	/* as above but wait for the laser disk to return 'R<CR>' */
	<i>ldpause port</i>	/* pause the laser disk at the given serial port, screen blanked */
	<i>wait for ldpause port</i>	/* as above but wait for laser disk to return 'R<CR>' */
	<i>wait for port idle</i>	/* wait for the give port to become idle, i.e. send buffer is empty */
	<i>if port idle in variable then jump label</i>	/* For Pioneer LD ports, also wait for pending command to return 'R<CR>' */
		/* This is a blocking call */
	<i>if port becomes idle in the given time, jump to label</i>	/* if port becomes idle in the given time, jump to label */
	<i>portlist:</i>	/* a single port or a comma separated list of ports */
	<i>port</i>	
	<i>portlist , port</i>	
	<i>stopbits:</i>	
	<i>one</i>	
	<i>one&half</i>	
	<i>two</i>	
	<i>wordsize:</i>	
	<i>5bits</i>	
	<i>6bits</i>	
	<i>7bits</i>	
	<i>8bits</i>	
	<i>parity:</i>	
	<i>none</i>	
	<i>odd</i>	
	<i>even</i>	

```
| high
| low
;
```

sexpression:

```
    asciistring
/    hexnumber          /* single byte hex numbers, e.g. 0x12 */
/    sexpression + sexpression
```

devicetype:

```
    midi
|    ld
|    dmx
|    ascii
|    int1
|    int2
|    int3
|    int4
;
```

tallytype:

```
    pwm
|    simple
```

line:

```
    line1
|    line2
```

isdayofweek:

```
    sunday          /* equivalent to a byte value of 0 */
|    monday        /* equivalent to a byte value of 1 */
|    tuesday       /* equivalent to a byte value of 2 */
|    wednesday     /* equivalent to a byte value of 3 */
|    thursday      /* equivalent to a byte value of 4 */
|    friday        /* equivalent to a byte value of 5 */
```

	saturday	/* equivalent to a byte value of 6 */
	<i>skey:</i>	
	skey (<i>number</i> , <i>number</i>)	/* specify a scanned key, for example skey(1,2), skey(3,4) etc */
	<i>extvar:</i>	
	extvar (<i>number</i> , <i>number</i>)	/* specify a port variable, for example portvar(1,2) */
	<i>expression:</i>	
	<i>timestring</i>	/* time strings are in the form of 12:34:56.12. Internally they are all converted to frames in long integer format */
/	<i>longint</i>	/* numbers followed by an 'I' or 'L' */
/	<i>number</i>	/* ordinary numbers. When used in expressions, they are treated as byte integers, i.e. between 0 and 255 */
/	<i>isdayofweek</i>	/* e.g. bvar = monday + 1; bvar would become 2. Or if bvar==tuesday then jump label; */
	<i>variable</i>	
/	now	/* this will push the current system time to the long/time stack, e.g. tvar = now + 0:0:1.0; if (now == 12:34:56.0) then jump label; */
	entrytime	/* this will push the sequence entry time to the long/time stack, e.g. tvar = entrytime + 0:0:1.0; */
	dayofweek	/* this will push the day of the week to the byte stack, e.g. if dayofweek == monday then jump domonday; However, this is equivalent to if monday then jump domonday; Similarly if dayofweek != monday then jump dontdomonday; is equivalent to if not monday then jump dontdomonday;

```

| true /* 1 */
| false /* 0 */
| on /* 1 */
| off /* 0 */
| open /* 1, be careful using this in expressions. opto open means 1 */
| closed /* 0, be careful using this in expressions, opto closed means 0 */
| tally
/ relay
/ opto
/ skey
| extvar
/ random(n) /* generates a random number between 0 and n /
/ elapsed /* elapsed time since beginning of sequence or since the time reference */
| elapsed(expression) /* elapsed time since relative to the time specified by the result of expression */
| vframe1 /* current video location of LD player at port 1 in frame count */
| vframe2 /* current video location of LD player at port 2 in frame count */
| vframe3 /* current video location of LD player at port 3 in frame count */
| vframe4 /* current video location of LD player at port 4 in frame count */
| vframe5 /* current video location of LD player at port 5 in frame count */
| vframe6 /* current video location of LD player at port 6 in frame count */
| vframe7 /* current video location of LD player at port 7 in frame count */
| expression + expression
/ expression - expression
/ expression * number
/ expression / expression
/ expression % expression
/ expression & expression
/ expression | expression
/ expression ^ expression
/ ~ expression
/ expression > expression

```

```

/* the result of this comparison is pushed onto the bit stack, so this is valid
bit bitvar;
bitvar = now > 12:34:56.0;
However, this is illegal

```

```

byte bvar;
bvar = now > 12:34:56.0;
*/

/  expression < expression
/  expression >= expression
/  expression <= expression
/  expression != expression
/  expression == expression
/  port == sexpression

/* compare the receive buffer of the port with the given ascii expression, e.g.
   if (port1 == "abc") then jump gotabc;
   if (port1 == "R" + 0x0D) then jump gotldreply;
   Buffer is untouched.
*/

/  port == number
/  port != number
/  port <= number
/  port >= number
/  port > number
/  port < number
/  ( expression )
;

opto:
|  opto1
|  opto2
|  opto3
|  opto4
|  opto5
|  opto6
|  opto7
|  opto8
|  opto9
|  opto10
|  opto11
|  opto12

```

relay:

```
relay1  
| relay2  
| relay3  
| relay4  
| relay5  
| relay6
```

tally:

```
tally1  
| tally2  
| tally3  
| tally4  
| tally5  
| tally6  
| tally7  
| tally8  
| tally9  
| tally10  
| tally11  
| tally12
```

port:

```
port1  
| port2  
| port3  
| port4  
| port5  
| port6  
| port7
```

asciistring: " [^\n]* "

/ Double-quoted sequence of characters except “ and linefeed */*

number: [0-9]+
/ one or more digits */*

longint: [0-9]+[iL]
/ one or more digits followed by ‘l’ or ‘L’ */*

hexnumber: 0[xX][0-9a-fA-F] [0-9a-fA-F]?
/ Single byte hex numbers. 0x or 0X followed by one or two hex digits */*

variable: [a-zA-Z_][a-zA-Z0-9_]*
/ Any string with a to z, A to Z or _. Leading character must be an alphabet */*

label: ^[a-zA-Z_][a-zA-Z0-9_]*[\t]*:
/ Must be at the beginning of the line otherwise similar to variables. Spaces and tabs allowed before the ‘:’ */*

timestring: ([0-1]?[0-9])|(2[0-3]):[0-5]?[0-9]:[0-5][0-9].[0-2]?[0-9]
/ anything of the form hour:minute:second.frame */*

Appendix B. Example Programs

1. Simple Video Loop

```
sequence VLoop
{
    time videoLen;
    time videoStart;
    videoLen = 12345L;           // 12345 frames long
    videoStart = 123L;          // start at location 123
    port1 type = LD;            // set port1 to type LD
    port1 baudrate = 4800;      // set baudrate to 4800
    reset port1;                // make sure port1 cleared
    wait for ldstart port1;     // spin up laser disk
```

```

loop:
    wait for ldsearch port1 to videoStart; // search to starting position
    idle for 30L;                          // wait for one second
    wait for ldplay port1;                  // start playing
    idle for videoLen;                      // wait until done
    jump loop;                              // rewind and play again
}

```

2. Synchronized Video Loop

```

sequence SyncLoop
{
    // play 3 videos synchronously
    time videoLen;
    time video1Start;
    time video2Start;
    time video3Start;
    videoLen = 1000L;                // 12345 frames long
    video1Start = 123L;              // video 1 start location
    video2Start = 1123L;             // video 2 start location
    video3Start = 2123L;             // video 3 start location
    port1 type = LD;                 // set port1 type to LD
    port2 type = LD;                 // set port2 type to LD
    port3 type = LD;                 // set port3 type to LD
    port1 baudrate = 4800;           // port1 baudrate is 4800
    port2 baudrate = 4800;           // port2 baudrate is 4800
    port3 baudrate = 4800;           // port3 baudrate is 4800
    reset port1;                     // make sure port1 cleared
    reset port2;                     // make sure port2 cleared
    reset port3;                     // make sure port3 cleared
    ldstart port1;                   // spin up laser disk 1 in the background
    ldstart port2;                   // spin up laser disk 2 in the background
    wait for ldstart port3;          // spin up laser disk 3 and wait until done
    wait for port1 idle;              // wait until laser disk 1 is done
    wait for port2 idle;              // wait until laser disk 2 is done too.
}

```

```

loop:
    wait for ldsearch port1 to video1Start;    // search to starting position
    wait for ldsearch port2 to video2Start;    // search to starting position
    wait for ldsearch port3 to video3Start;    // search to starting position
    idle for 30L;                             // wait for 1 second
    hold port1,port2,port3;                   // hold ports
    ldplay port1;                             // issue play command
    ldplay port2;                             // issue play command
    ldplay port3;                             // issue play command
    framesync;                                // wait until beginning of next video frame so that players start within the same frame.
                                                // This is meaningful only if the controller and the laser disk players are sharing the same
                                                // external video frame sync signal.

    release port1,port2,port3;                // let go of the players
    idle for videoLen;                         // wait until done playing
    jump loop;                                // back to beginning of the loop
}

```

3. Video On Demand

```

sequence OnDemand
{
    // a video is played when a button is pushed
    time video Len;
    time videoStart;
    videoLen = 12345L;                         // 12345 frames long
    videoStart = 123L;                         // start at location 123
    port1 type = LD;                           // set port1 to type LD
    port1 baudrate = 4800;                     // set baudrate to 4800
    reset port1;                               // make sure port1 cleared
    wait for ldstart port1;                    // spin up laser disk
loop:
    wait for ldsearch port1 to videoStart; // search to starting position
    tally1 = on;                              // turn on tally light for attraction
    clear opto1 down;                          // clear any pending button push. Remove this line if you want button push remembered.
    wait for opto1 down;                       // wait until there is a button push
}

```

```
tally1 = off;           // turn off tally light during video
wait for ldplay port1; // start playing
idle for videoLen;     // wait until done
jump loop;             // rewind and wait
}
```