RevPower SCSL™ (Sign Control Scripting Language) Release 1.9

Document Revision 1.4 16-Aug-2000 JDS

```
Mxxx'       change machine ID in next packet (0 - 127).
Lxxx'       change lamp level in next packet (0 - 100 %).
Sxxx'       change transport speed in next packet (0 -  100 %).

Wxx.x'      pause for x seconds (resolution 0.0001, min. 0.02).
Hxx:xx:xx   wait until the specified 24-hour time.
Kxx         wait for any key or ASCII key xx.

Fxx'        go to frame xx using setup options (1 - 32).
Txx'        go to frame xx using sign's internal table data (1 - 32).
Pxx'        program frame xx in sign's internal table data (1 - 32).
Ixxx'       force lamp illumination to level xxx (0 - 100 %).
G           get sign status for current machine ID.
Zxx         execute special function code (0 - 31).
X           STOP the currently specified machine.

Rxxx        repeat command line x times or until all values settle.
'xxx        specify repeat step value for applicable command codes.

READ        open and execute the command lines in a file.
LOOP        reload and repeat the current file (endless loop).
QUIT        close all open files and quit the interpreter program.
```

The Revolution Sign Control Scripting Language (SCSL) is a sophisticated control language for determining the actions and behavior of Revolution multi-frame display signs. This protocol is supported by **RevPower™** PC software.

Control Structure

There are three control structure tiers to the SCSL protocol:
1. **Command codes** consisting of a letter and associated numeric values.
2. **Command lines** are a series of command codes terminated with the ENTER key.
3. **Files** consist of a series of command lines and are saved in standard PC text format.

Files can contain an unlimited number of command lines. Files can include directives to load other files or continuously repeat the current file. Files must terminate with an END or LOOP statement. If the command interpreter encounters the end of a file without an END or LOOP statement, an error will occur.

A **command** line can be a string of up to 256 characters. A typical **command code** consumes two or three characters. No spaces are required between command codes but can be included to improve readability. A command line can use the READ directive to load and process a file. As is typical for text files, a command line is terminated with a carriage-return character (the ENTER key).

The SCSL 1.x protocol supports file nesting but not file recursion. That is, a file can be opened that includes the READ directive to load another file. That newly opened file can again READ another file. This is called “file nesting”. If a READ command tries to open a file that is already open in the nested sequence, an error will occur; this is “file recursion” and is not supported by SCSL 1.x.

Basic Commands

All command codes have three basic parts, **letter**, **value**, and **step**, in the following syntax:

```
L26'2
```

The STEP value is preceeded by the tick (') symbol and is optional.
Thus, the most basic syntax does not include the STEP value:

```
L26
```

In these examples, the letter is **L** and the value is **26**. In the first example, a STEP value of **2** is also specified. There are fourteen valid letter codes, each with their own range of valid values. These letter codes consist of three types:

Data setup codes:

```
Mxxx'       change machine ID in next packet (0 - 127).
Lxxx'       change lamp level in next packet (0 - 100 %).
Sxxx'       change transport speed in next packet (0 - 100 %).
```

Data output codes:

```
Fxx'        go to frame xx using setup options (1 - 32).
Txx'        go to frame xx using sign's internal table data (1 - 32).
Pxx'        program frame xx in sign's internal table data (1 - 32).
Ixxx'       force lamp illumination to level xxx (0 - 100 %).
G           get sign status for current machine ID (when supported).
Zxx         execute special function code (0 - 31).
X           STOP the currently specified machine.
```

Timing control codes:

```
Wxx.x'      pause for x seconds (resolution 0.0001, min. 0.02).
Hxx:xx:xx   wait until the specified 24-hour time.
Kxx         wait for any key (if no value specified) or ASCII key  xx.
```

Flow control codes:

```
Rxxx        repeat command line x times or until all va lues settle.
'xxx        specify repeat step values for applicable command codes.
```

Data setup codes set various parameters for the next outgoing control packet. They do not immediately result in a motion or change. These commands include **M** to set the machine ID of the next packet, **L** to set the lamp level in the next packet, and **S** to set the speed in the next packet. Valid machine IDs are from 0(zero) to 127. M0 will always control all signs at the same time. L and S values are percentages from 0(zero) to 100 percent. A lamp level of 0 turns the lamp circuit off; a level of 1 provides the lowest possible dimmer setting without switching off (different sign models may provide different minimum levels).

Data output codes transmit control information to physical signs. The sign or signs being controlled are always determined by a previously set M command (if no M has been specified then M0 will be assumed, which controls all signs simultaneously). Primary commands are **F** to move to a frame using previously set lamp and speed values (via L and S commands), **T** to move to a frame using the sign's internal lamp and level tables, and **P** to program the values of L and S into the sign's internal table. Frame number 1 is the top frame in a physical sign. Frame 32 is the lowest possible frame. If a physical sign has less than 32 spaces available, F and T commands will only work to the last real frame. P will always work over the full range of 1 to 32.

Lamp levels set with the **L** command will take effect after the sign completes its next motion (the new level is activated in the sign when stopping at its destination). This command is a data setup code.

Lamp level can be changed immediately with the **I** command. This command is a data output code.

Other data output codes include **G** to return the status of a sign (provided the connection hardware is bidirectional), and **Z** to send a special function command (discussed later in this document).

Timing control commands determine when an event will occur.

W causes a delay in seconds before executing the next command. This is the only SCSL 1.x command that can process floating-point numbers – numbers with a decimal point, like 6.125 or 0.1116. (Other commands will round floating-point numbers to integers before processing).

H waits until a specified 24-hour time. This is the only SCSL 1.x that uses twenty-four clock time as it's value. No step value can be specified for this command. For example, `H15:30:00` causes the system to wait until 3:30pm before proceeding with the next instruction. If the time is presently 3:31pm (or later), the system will wait up to twenty-four hours for the clock to loop back to 3:30pm again. The time reference is the PC clock and should be set correctly for expected results.

K waits for a key to be pressed on the control computer. If no value is specified the system will wait for any key to continue or the ESC key to close all open files and return to the interpreter command line. If a value is specified for K, the system will wait for the key that produces that ASCII value. For example, the spacebar is ASCII32, the ESC key is ASCII27. If K27 is specified, the ESC key will cause the system to continue as intended, not close files and return to the command line. Upper- and lower-case letters have different ASCII codes and some PC keys do not have standard ASCII values.

> ***Hint: If you are reading this to learn SCSL programming, we suggest running the command line interpreter program now and experimenting with the various command codes described so far. An intuitive understanding of these basic codes is necessary to fully utilize the power of the flow control commands described next.***

Flow control codes allow multi-command sequences to repeat and self-modify to achieve complex results. The **R** command creates a repeat-loop for all commands following it on the command line (the loop terminates at the line end). Several of the packet configuration commands and packet send commands allow **STEP** values to be specified with the tick(') symbol. This causes the value of that command to be incremented by the STEP value each time the line repeats. Values will only increment until the maximum or minimum value allowed for that parameter is reached. For example, stepping the L level by 5 will increment the lamp brightness by 5% on each repeat until the lamp reaches 100%. If the value increments higher than the maximum, it is reduced to the maximum and the STEP value is cleared to zero. STEP values can be positive or negative and are usually integers. Fractional values (floating-point numbers like 1.625) will be rounded to the nearest integer when required by the command. As of SCSL 1.x, the only command that can use a fractional value is W for time delays in seconds.

If the R command has a value, the subsequent commands on the line will repeat that many times. For example, the line `R4 M1'1 F1` will repeat four times, incrementing the M value each time. The speed of these messages is very fast, so the result will appear to be signs 1, 2, 3, and 4 all moving to frame 1 simultaneously while other signs remain where they were.

If the R command does not have a value, the line will repeat until all values have reached their minimum or maximum values and all STEP values have been cleared to zero. For example, R I10'10 will repeat until the value of I has reached 110 in steps of 10. The value 110 is not allowed because the maximum percentage value for I is 100%. Thus, on the 10th repeat of the R command, the I level is reduced to 100, the STEP value is cleared, and repeating stops.

File directives determine how the system handles text files. Directives must appear on a line by themselves.

READ *filename* loads the specified file and executes each command line in the file in sequential order. The *filename* must exist in the PATH directory. Example: `READ BIGSHOW` or `READ BIGSHOW.TXT`. As a shortcut, the dash (-) character can be substituted for the directive, i.e. `-BIGSHOW.TXT`.

PATH (by itself, without a pathname) will report the current path where the interpreter is looking for files.

PATH *pathname* sets the path as specified. This should be a complete path starting with a root drive letter, i.e. `PATH c:/revpower/`.

LOOP causes the current file to be reloaded from the beginning and run again (commands after the LOOP directive will never be executed). This directive should not be the first or only command line of a file.

END or **LOOP** is required at the end of SCSL 1.x instructions to ensure proper handling of "nested" file **READ**s. Any number of lines may follow the **END** statement but will be ignored by the command interpreter.

Within a text file, the # symbol is used to delimit comments from commands. The system will ignore all characters after the **#** on any line. Lines that begin with **#** will be completely ignored.

Special Function Codes

SCSL 1.x special function codes include the following:

```
Z1    STOP (the X command invokes this special function)
Z2    Learn marks for all frame positions
Z3    Reset and restart machine
Z4    Find the top frame and realign position sensors
```

Z0 and codes up to Z31 are reserved for special functions, including internal standalone sequence uploading and system setup. These codes are used by other PC applications (including **RevSeq™** and **RevSetup™**) and should normally be avoided in SCSL **RevPower™** sequences.

Commenting SCSL 1.0 Scripts

Comments are an important part of any programming or scripting language to ensure readability and easier editing later. SCSL 1.x supports two different kinds of comments.

Inline comments must begin with the # symbol and can be anywhere in the script. All characters from the # to the end of the line will be ignored by the command interpreter. For example, `M1 F2 # M2 F3` will only send machine 1 to frame 2. The rest of the line is commented out with the #. A more typical use would be `M0 F5 # Send all machines to Coke ad in frame 5`. Use comments to help clarify the code you have written.

It is typical for the first few lines of a script to specify the filename, a short explanation of the script, the author's name, the date written, a version number, etc:

```
# 10TOP.TXT
# Move signs 1 – 10 to top frame
# James David Smith, 06-Jan-2000
# Ver 1.0
#
R10 M1'1 F1 # repeat M1 through M10 going to top frame
END         # all SCSL 1.0 files MUST terminate with END or LOOP
```

File end comments can be freely added after the END or LOOP command. The command interpreter never loads lines after END or LOOP, so the # is not required and all characters are legal. (You could even put HTML documentation here!)

Examples and Explanations

Command line examples:

`R100 M6 I1'1 W0.1`
This line will repeat 100 times.
Machine 6 is selected.
The lamp level is forced to 1 (lowest dimmer setting).
On each repeat the I command is incremented (stepped up) by 1.
Each repeat occurs after 1/10 of a second.
Result: the lamp on machine six slowly dims from minimum to maximum over ten seconds.

`R M0 F1'2 W5`
This line will repeat until values can no longer step.
All machines are selected (M0).
All machines go to frame 1 using previously defined L and S settings.
On each repeat the frame number will increment by 2.
Each repeat occurs after a 5 second delay.
Result: all machines start at frame 1 then move to 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 32. Motion ends at 32 because that is the highest possible value. There is a 5-second pause between each change. To stop at frame 31, change the `R` to `R15`.

`R M0 F31'-2 W5`
The opposite of the previous line. All signs move to frame 31 then step down two frames every 5 seconds, ending on frame 1.

`R19 M1'1 T1 W0.2`
This line will repeat nineteen times.
Machine 1 is selected. On each repeat the machine number will increment by 1.
The specified machine will go to frame 1 using its internal table for speed and lamp level.
There will be a pause of 2/10 of a second before repeating the line.
Result: machines 1 to 20 change to frame 1 in a chase pattern over 2 seconds.

Text file examples:

```
#
# Characters after the # symbol are ignored.
# Lines that start with # are skipped.
# This is a sample Revolution SCSL file named CYCLE.TXT
#
M0
R T1'1 W10
R T32'-1 W10
LOOP
```

With only four lines of script this file causes all signs to cycle from the first frame to the last frame, back to the first frame, and repeat forever. Each frame is displayed for 10 seconds except for the first and last frames which get 20 seconds.

```
#
# Program default table: PROGRAM.TXT
#
M0 L100 S100  # set packet levels to maximum
R P1'1        # step through all frame spaces
END
```

This file is directed at all connected machines via the M0 command.
It first sets packet values for a speed of 100% and a lamp level of 100%.
It then programs these values into the sign's internal table for all frames from 1 to 32.
The R command stops when the last frame has been programmed.
This file also demonstrates the use of comments in command lines.

```
#
# Factory test sequence TEST.TXT
#
READ PROGRAM.TXT
READ CYCLE.TXT
END
```

This file illustrates the "nesting" power of SCSL.
First, the file PROGRAM.TXT is loaded and run, setting all table values to 100% in all signs. Then the file CYCLE.TXT is loaded and run, which cycles through all frames forever. Because of the LOOP directive in CYCLE.TXT, the interpreter never returns to this parent file and the END directive is never read. Nonetheless, it is good programming practice to always include an END or LOOP terminator in SCSL script files.