

WEM|Script™

Getting Started with ProCommander® Scripting

An introductory guide to sending commands, starting shows, working with variables, and controlling digital and analog outputs.

Scope This guide focuses on the most common WEM|Script™ concepts and commands used when first working with a ProCommander®. It is not a complete command reference.

Version 090226

1. Overview

ProCommander® show controllers are most often used to play pre-built shows created in a show-production application such as Showforge®. WEM|Script™ adds another layer of control by allowing you to send instructions directly to the ProCommander®, respond to triggers and system states, and place commands inside a show timeline.

This guide introduces the basic command format and several common tasks: starting and stopping shows, reading and changing variables, controlling digital outputs, and setting or fading analog outputs.

Full command reference The latest published list of WEM|Script™ commands is available on the Support page of the Weigl Controls website under Product Manuals.

2. Where WEM|Script™ Can Be Used

Control.INI file

The control.ini file is an ASCII text file placed in the root directory of the ProCommander® memory card or, on models that support it, onboard memory. Each line contains a WEM|Script™ action preceded by one or more trigger or check statements. The file is scanned when the memory is inserted or when the device powers on, and the ProCommander® continuously evaluates the configured conditions.

This makes control.ini useful for autonomous behavior such as reacting to an input, waiting for a timer, or performing an action when a variable reaches a specific value. Complete control.ini examples and training material are available on the Weigl Controls Support page.

Direct UDP command

WEM|Script™ commands can be sent directly over the network as UDP messages. Send commands to one of the three configurable UDP ports on the ProCommander® (default: 5555) or to system port 22202. For commands that return a response or acknowledgement, the device sends that response back to the source port that originated the message.

For quick testing, the console in the Weigl Hardware Configurator can also send UDP commands to a connected device.

Embedded in WM1 shows

A show-authoring application such as Showforge® can place WEM|Script™ commands directly into a WM1 show. During playback, each command is executed at the exact frame where it was cued on the show timeline.

3. WEM|Script™ Command Format

Basic syntax

Every command begins with an exclamation point and ends with a number sign:

!<command text>#

Compound commands

A command can be prefixed with one or more triggers or checks. These prefixes act as conditions that must be satisfied before the final action is executed.

```
!<check variable 1 = 1>!<check variable 2 = 1>!<start show>#
```

Important A statement can contain multiple triggers or checks, but only one action at the end.

Triggers vs. checks

Trigger: A special prefix used only in control.ini statements. Many triggers are event- or time-based, such as an action that occurs a set time after power-up. Trigger commands are listed near the beginning of the published WEM|Script™ ASCII command document.

Check: A conditional prefix that can be used anywhere WEM|Script™ commands are accepted. Checks usually validate a current state—such as a variable value—before allowing the final command to execute.

4. Starting and Ending Shows

WEM|Script™ provides several versions of the show-start command. The correct choice depends on what should happen if another show—or the same show—is already running. In the examples below, X represents the show number.

Operation	Command	What it does
Normal start	<code>!rsnX#</code>	Start show X only when no other show is currently running.
Add start	<code>!rsaX#</code>	Start show X without interrupting any shows that are already running.
Interrupt start	<code>!rsiX#</code>	Stop all running shows with a different number than X. If show X is already running, stop it and restart it from the beginning.
Terminate start	<code>!rstX#</code>	Stop all running shows with a different number than X. If show X is already running, let it continue; otherwise start it from the beginning.
Restart show	<code>!rstX#</code>	If show X is not running, start it. If it is already running, restart it from the beginning. Other shows are not affected.
Start show number from variable	<code>!rsvnX#</code> <code>!rsvaX#</code> <code>!rsviX#</code> <code>!rsvtX#</code> <code>!rsvrX#</code>	Same behaviors as the standard start variants, except X is the variable number containing the show number to use.
End show	<code>!rseX#</code>	Stop show X if it is running. Use X = 0 to stop all shows.

`!rsn12#` Start show 12 only if no other show is running.

`!rse0#` Stop all currently running shows.

5. Variables

The ProCommander® includes numbered internal variables that can each hold a 16-bit value from 0 to 65535. Variables are useful for counters, state flags, selections, and many other scripting tasks.

Variable range	Storage	Behavior
1–70	Volatile	Reset to 0 each time the device powers on.
71–100	EEPROM	Retain their values through a power cycle. Out of the box, these variables default to 65535.

EEPROM write-cycle note EEPROM variables have a finite write-cycle life. They are appropriate for values that need to survive power cycles—such as a show run count—but should not be used for rapid timers, tight loops, or logic that may write hundreds or thousands of times per second.

Variable command reference The commands in this section are only a few of the most commonly used variable commands. The complete variable command set is available in the full WEM|Script™ ASCII command reference on the Weigl Controls Support page under Product Manuals.

Purpose	Command	Details
Set using a constant	<code>!vmcX<operator>Y#</code>	X = destination variable; operator = =, +, -, *, or /; Y = constant value. Result is clamped to 0–65535.
Set using another variable	<code>!vmvX<operator>Y#</code>	X = destination variable; operator = =, +, -, *, or /; Y = source variable number. Result is clamped to 0–65535.
Read one variable	<code>!gvcX#</code>	Returns VarX=<value>.
Read all variables	<code>!gva#</code>	Returns ASCII Var: followed by the complete variable contents in hexadecimal, 2 bytes per variable (high byte, low byte), terminated by <CR><LF> (0D 0A).
Check variable against a constant	<code>!vccX<operator>Y!<command>#</code>	Prefix that compares variable X with constant Y. Operators: =, >, <, or ~ (not equal). If the condition is true, the following command is executed. Only one command may follow the check.

!vmc1=5# Set variable 1 to 5.

!vmc1+1# Add 1 to variable 1.

!gvc1# Read the current value of variable 1.

!vcc5>0!rsn1# Start show 1 in normal mode only if variable 5 is greater than 0.

6. Setting Digital Output Values

Most ProCommander® models include digital open-collector outputs. The commands below can clear outputs, set outputs, explicitly define the on/off state of a group of outputs, and mask selected outputs from later PC or show commands.

How the output pattern works

Digital output patterns are left-aligned. The first character represents output 1, the second represents output 2, and so on, up to 16 outputs. If fewer than 16 characters are supplied, outputs beyond the end of the pattern are not addressed and therefore do not change.

Key distinction For !eas and !eos, a 0 means “do not affect this channel.” For !ess, a 0 means “clear this channel.” This difference is important.

Purpose	Syntax	Pattern meaning
Clear all outputs	!eao#	Sets every digital output to 0.
Clear selected outputs	!eas<pattern>#	1 = clear this output; 0 = leave this output unchanged. Unaddressed outputs remain unchanged.
Set selected outputs	!eos<pattern>#	1 = set this output; 0 = leave this output unchanged. Unaddressed outputs remain unchanged.
Set explicit output states	!ess<pattern>#	1 = set this output; 0 = clear this output. Unaddressed outputs remain unchanged.

Examples: clear selected outputs (!eas)

Command	Result
!eas1001#	Clear outputs 1 and 4. Outputs 2 and 3 are unchanged; outputs 5–16 are not addressed.
!eas10011#	Clear outputs 1, 4, and 5. Outputs 2 and 3 are unchanged; outputs 6–16 are not addressed.
!eas0001#	Clear output 4 only. Outputs 1–3 and 5–16 are unchanged.
!eas1111111111111111#	Clear all 16 outputs.
!eas0000000000000000#	Make no changes to any of the 16 outputs.
!eas1111000001000101#	Clear outputs 1, 2, 3, 4, 10, 14, and 16; leave the others unchanged.

Examples: set selected outputs (!eos)

Command	Result
!eos1001#	Set outputs 1 and 4. Outputs 2 and 3 are unchanged; outputs 5–16 are not addressed.
!eos10011#	Set outputs 1, 4, and 5. Outputs 2 and 3 are unchanged; outputs 6–16 are not addressed.
!eos0001#	Set output 4 only. Outputs 1–3 and 5–16 are unchanged.
!eos1111111111111111#	Set all 16 outputs.
!eos0000000000000000#	Make no changes to any of the 16 outputs.
!eos1111000001000101#	Set outputs 1, 2, 3, 4, 10, 14, and 16; leave the others unchanged.

Examples: set explicit states (!ess)

Command	Result
<code>!ess1001#</code>	Set outputs 1 and 4; clear outputs 2 and 3. Outputs 5–16 are not addressed.
<code>!ess10011#</code>	Set outputs 1, 4, and 5; clear outputs 2 and 3. Outputs 6–16 are not addressed.
<code>!ess0001#</code>	Set output 4; clear outputs 1, 2, and 3. Outputs 5–16 are not addressed.
<code>!ess1111111111111111#</code>	Set all 16 outputs.
<code>!ess0000000000000000#</code>	Clear all 16 outputs.
<code>!ess1111000001000101#</code>	Set outputs 1, 2, 3, 4, 10, 14, and 16; clear outputs 5, 6, 7, 8, 9, 11, 12, 13, and 15.

Masking digital outputs

Digital masks control whether later commands from the PC or a running show are mapped to each output. Mask patterns support up to 16 outputs and are left-aligned: the first character is output 1. Use 1 for not masked, 0 for masked, and x to leave the existing mask state unchanged. Outputs beyond the end of the pattern are not changed. Write the pattern continuously from output 1; use x when you need to preserve a channel in the middle of the pattern.

Command	Behavior	Example
<code>!edm<mask>#</code>	Set or change the digital mask without forcing the newly masked outputs off. Masked outputs no longer receive commands from the PC or show, allowing direct input control regardless of show content.	<code>!edm1001#</code> Outputs 2 and 3 are masked; 1 and 4 are not masked; outputs 5–16 are unchanged.
<code>!emo<mask>#</code>	Set or change the digital mask and switch every masked channel off. Masked outputs no longer receive later mapped commands.	<code>!emo1001#</code> Outputs 2 and 3 are masked and switched off; 1 and 4 are not masked; outputs 5–16 are unchanged.

Using x in a mask `!edm0x0x0000000000#` masks outputs 1, 3, 4, and 6–16 while leaving the current mask state of outputs 2 and 5 unchanged.

7. Setting Analog Output Values

The number and type of analog, R/C-servo, and CAN outputs depend on the ProCommander® model. WEM|Script™ can set an output to a value immediately, fade it to a target over time, and mask selected analog outputs.

Standard and Enhanced Command Formats

The analog set and fade commands are available in two addressing formats: standard and enhanced. The standard format is the normal format used for physical analog and R/C-servo outputs and is the format most users will need. The enhanced format is available on devices that support separate CAN addressing, allowing CAN channels to be addressed independently from the physical analog and R/C-servo outputs. Support varies by model; check the documentation for your specific ProCommander®.

Standard format: uses a single output address X. Examples: `!eslX%Y#` and `!eflX%Y<Z#`.

Enhanced CAN format: uses W.X addressing, where W selects a supported physical CAN port and X selects the channel on that port. Examples: `!eslW.X%Y#` and `!eflW.X%Y<Z#`.

Most users The standard command format is all that is required for typical analog and R/C-servo control. The enhanced CAN format applies only to devices that support separate CAN addressing and requires separate CAN addressing mode, enabled with `!onc1#`. Check the documentation for your specific device, and see Analog and CAN Addressing Modes later in this section for details.

Parameters Used in the Analog Commands

The command tables below use parameter letters to show exactly where each value belongs in the complete command. The exact ranges and operators available for each parameter are shown in the command tables that follow.

Parameter	Used in	Meaning
W	Enhanced W.X format	CAN port selector. Used only with the enhanced command format. W selects a physical CAN port supported by the device (for example, port 1 or port 2 on a device that provides two CAN ports).
X	Standard and enhanced formats	Output/channel selector. In the standard format, X selects the analog, R/C-servo, or other supported output address. In the enhanced W.X format, X selects CAN channel 1-16 on the CAN port selected by W.
Y	Level commands	Target value. % supplies 0-100%; = supplies a raw 10-bit value from 0-1023. On supported 16-bit command forms, : supplies 0-65535. The <code>!esl</code> command also supports + and - for relative changes.
Z	<code>!efl</code> fade commands	Fade time, up to 600 seconds. With <, write the time directly in seconds to a tenth (for example <code><2.5</code> = 2.5 seconds). With >, enter tenths of a second (for example <code>>25</code> = 2.5 seconds).

Fade to a Level Over Time (!efl)

Use `!efl` when an output should move from its current value to target Y over time Z. The standard form uses address X; the enhanced CAN form uses W.X.

Use	Full command format	Notes
Standard - percent	<code>!eflX%Y<Z#</code> or <code>!eflX%Y>Z#</code>	Standard addressing; Y = 0-100%.
Standard - 10-bit	<code>!eflX=Y<Z#</code> or <code>!eflX=Y>Z#</code>	Standard addressing; Y = 0-1023.
Enhanced CAN - percent	<code>!eflW.X%Y<Z#</code> or <code>!eflW.X%Y>Z#</code>	W = CAN port; X = CAN channel. Requires separate CAN addressing mode.
Enhanced CAN - 10-bit	<code>!eflW.X=Y<Z#</code> or <code>!eflW.X=Y>Z#</code>	Enhanced CAN addressing; Y = 0-1023.

Use	Full command format	Notes
Enhanced CAN - 16-bit	!eflW.X:Y<Z# or !eflW.X:Y>Z#	Where supported; Y = 0-65535.
Stop an active fade	!eflX #	Stops the active !efl fade on standard address X.

Examples

Example command	Explanation
!efl1%20<2.5# or !efl1%20>25#	Fade standard output address 1 to 20% in 2.5 seconds.
!efl1_4%50<20.8# or !efl1_4%50>208#	Fade standard output addresses 1 through 4 to 50% in 20.8 seconds.
!efl3=720<200# or !efl3=720>2000#	Fade standard output address 3 to raw 10-bit level 720 in 200 seconds.
!efl5_8%0<1# or !efl5_8%0>10#	Move R/C-servo outputs 1 through 4 to the 0-position in 1 second.
!efl3 #	Stop the active fade on standard output address 3.
!efl1.1%50<2.5#	Fade CAN port 1, channel 1 to 50% in 2.5 seconds. Requires separate CAN addressing mode.
!efl2.6:35000<3#	Fade CAN port 2, channel 6 to 16-bit level 35000 in 3 seconds where the 16-bit form is supported. Requires separate CAN addressing mode and a device that provides CAN port 2.
!efl1.9=560<7#	Fade CAN port 1, channel 9 to raw 10-bit level 560 in 7 seconds. Requires separate CAN addressing mode.

Set a Level Immediately (!esl)

Use !esl when the output should change immediately rather than fade. The standard form uses address X; the enhanced CAN form uses W.X.

Use	Full command format	Notes
Standard - percent	!eslX%Y#	Set standard address X to 0-100%.
Standard - 10-bit	!eslX=Y#	Set standard address X to raw level 0-1023.
Standard - relative	!eslX+# !eslX-# !eslX+Y# !eslX-Y#	Increment/decrement by 1 or by the supplied amount; clamped to the supported range.
Enhanced CAN - percent	!eslW.X%Y#	W = supported CAN port; X = CAN channel 1-16. Requires separate CAN addressing mode.
Enhanced CAN - 10-bit	!eslW.X=Y#	Set the selected CAN channel to 0-1023.
16-bit where supported	!eslX:Y# or !eslW.X:Y#	Set the selected standard or enhanced CAN address to 0-65535 where the 16-bit form is supported.

Examples

Example command	Explanation
!esl1%20#	Set standard output address 1 to 20% (approximately 2 V on a 0-10 V range).

Example command	Explanation
!esl1_4%50#	Set standard output addresses 1 through 4 to 50% (approximately 5 V on a 0-10 V range).
!esl3=720#	Set standard output address 3 to raw 10-bit level 720 (approximately 7 V on a 0-10 V range).
!esl7%30#	Set R/C-servo output 3 to the 30% position.
!esl5_8%0#	Set R/C-servo outputs 1 through 4 to the 0-position.
!esl2+# / !esl2-#	Increment or decrement standard output address 2 by 1.
!esl6+10# / !esl6-20#	Add 10 to or subtract 20 from the current value of standard output address 6.
!esl1.1%50#	Set CAN port 1, channel 1 to 50%. Requires separate CAN addressing mode.
!esl1.1:65535#	Set CAN port 1, channel 1 to 65535 using the 16-bit form where supported. Requires separate CAN addressing mode.
!esl1:40000#	On hardware supporting the 16-bit standard form, set standard output address 1 to 40000.

Analog and CAN Addressing Modes

For most applications, the standard analog command format described above is all that is required. Some ProCommander® devices support separating CAN addressing from physical analog and R/C-servo addressing. On supported devices, the !onc command selects whether the device uses traditional legacy mirrored addressing or separate CAN addressing. Support varies by model; check the documentation for your specific device.

Legacy Mirrored Addressing (!onc0#)

When !onc0# is active, the device uses legacy mirrored addressing. The standard command format is used and physical analog/R/C-servo outputs share their addresses with CAN port 1. Values sent to those standard addresses are mirrored to both the corresponding physical output and the corresponding CAN channel.

- Standard address 1 controls physical analog output 1 and CAN port 1, channel 1.
- Standard address 4 controls physical analog output 4 and CAN port 1, channel 4.
- Standard address 5 controls R/C-servo output 1 and CAN port 1, channel 5.
- Standard address 8 controls R/C-servo output 4 and CAN port 1, channel 8.
- Standard addresses 9-16 address CAN port 1 channels 9-16.

The enhanced W.X command format is not available in this mode, and CAN channels cannot be addressed separately from the corresponding physical output addresses.

Separate CAN Addressing (!onc1#)

Sending !onc1# enables separate CAN addressing on hardware that supports it. In this mode, physical analog and R/C-servo outputs continue to use the standard command format, but they are no longer mirrored to CAN. CAN channels are instead controlled independently using the enhanced W.X format.

!esl1%50# addresses physical analog output 1 only.

!esl1.1%50# addresses CAN port 1, channel 1.

!esl2.1%50# addresses CAN port 2, channel 1 on devices that provide a second CAN port.

This allows the physical outputs and supported CAN ports to be controlled independently without sharing output addresses.

Choosing a Mode

Before changing the addressing mode, confirm that your device supports separate CAN addressing by checking its documentation. You can check the current mode by sending !onc# (with no 0 or 1) or by viewing the setting in the Weigl Hardware Configurator application.

Use !onc0# when legacy mirrored addressing is desired. Use !onc1# when you need to address the device's physical analog/R/C-servo outputs and CAN channels independently.

Important when changing modes Changing !onc changes how output addresses are interpreted. Existing !efl, !esl, !sao, and other logic containing output-channel assignments may need to be reformatted when switching between legacy mirrored addressing and separate CAN addressing.

Devices that do not support separate CAN addressing use the standard command format and do not use the enhanced W.X addressing scheme.

Masking analog outputs

The !eam command applies an analog mask to as many as 16 outputs. The mask is left-aligned: the first character is output 1. Use 1 for not masked, 0 for masked, and x to leave the current mask state unchanged. Outputs beyond the end of the pattern are unaffected. Write the pattern continuously from output 1; use x to preserve a channel in the middle. Every channel that becomes masked is switched off, and no further mapped commands are sent to that masked output.

Command	Result
!eam1001#	Outputs 2 and 3 are masked and switched off. Outputs 1 and 4 are not masked. Outputs 5-16 are not affected. Further mapped commands are no longer sent to outputs 2 and 3.
!eam0x0x0000000000#	Outputs 1, 3, 4, and 6-16 are masked and switched off. The current mask state of outputs 2 and 5 is unchanged.

8. Recommended Next Steps

Once you are comfortable with the command format and the examples in this guide, the next step is to work through the full WEM|Script™ ASCII command reference and the control.ini examples on the Weigl Controls Support page. Those resources cover the larger command set, trigger types, checks, device-specific functions, and more advanced scripting behavior.

Training tip When learning a new command, test it first with a simple, observable result—such as starting a test show, changing a variable, or toggling one output—before combining it with triggers or checks.