Cpc Script Language (CSL)

CSL is a scripting language that allows you to drive an emulator quite precisely, replacing user actions. CSL version 1.0 contains the following instructions:

csl_version <version>	Indicates the version of the CSL format to the emulator The most recent version is version 1.0
	Example : csl_version 1.0
reset <soft | hard (default)> reset	Reset the emulator. 'soft': memory cleared by the rom and only concerns the 64K of central ram. 'hard': all the ram is cleared
	Example : reset soft
crtc_select <num_crtc>	Selects the emulated CRTC. num_crtc can be 0, 1, 2, 3, 4 as well as the values 1A, 1B or PUSSY (1B corresponds to 1).
	Example : crtc_select 2
disk_insert <drive> <'file with extension'>	Insert a file into the specified drive If <drive> is not specified, drive A is considered by default.
	Example : disk_insert 'SHAKER25.DSK' disk_insert B 'AMAZING.DSK'
disk_dir <'disk directory'>	Specifies the DSK directory name concatenated before the disk name. If disk_dir is not used, the current emulator directory for DSKs is used.
	Example : disk_dir 'C:\MyDsk\'
tape_insert <'file with extension'>	Inserts a file into the tape player.
	Example : tape_insert 'MyTape.CDT'
tape_dir <'tape directory'>	Specifies the CDT directory name concatenated before the tape name. If tape_dir is not used, the current emulator directory for CDTs is used.
	Example : tape_dir 'C:\MyTapes\'
tape_play	Start playback of the last tape inserted via tape_insert.
	Example : tape_play
tape_stop	Stops playing the last tape inserted via tape_insert.
	Example : tape_stop
tape_rewind	Rewinds the last tape inserted via tape_insert to the beginning.
	Example : tape_rewind
snapshot_load <'snapshot file with extension'>	Loads a snapshot.
	Example :

	snapshot_load 'boink.sna'
snapshot_dir <' snapshots directory'>	Specifies the SNA directory name concatenated before the Snapshot file name. If snapshot_dir is not used, the current SNA emulator folder is used.
	Example : snapshot_dir 'C:\MesSNA\'
key_delay <key press delay in µsec> <delay in µsec between 2 keys><delay in µs after a CR code>	Sets the key press delay in µsec, the waiting speed between 2 keys in µSec, and the time in µsec after sending a CR (Carriage Return). The second parameter is optional (in this case, the delay after CR is identical to that of another key)
	Example : key_delay 500000 1000000 Délai 0.5sec between 2 keys, and 1 sec after CR
key_output <'Text' >	Sends one by one the characters of the string 'Text' passed as an argument. If a character is unknown, send nothing. The sending of a special character follows the following format: \(code) (see table in appendix) The simultaneous sending of two keys is achieved by putting the characters between 2 braces: {abcd} The delay between characters is specified with key_delay. By default, it is 19968 µsec. The delay (defined with key_delay) between two keys (or group of keys) is ignored if the first key (or first group) is followed by the character \(KOF)
	Example : key_output 'RUN "SHAKE25A"\(RET)' key_output '{\(SHI)1} >> SHIFT + 1 keys
key_from_file <'ascii file'>	Sends one by one the characters contained in the file whose name is passed as an argument. If a character is unknown, send nothing. The delay between characters is specified with key_delay. By default, it is 19968 µsec.
	Example : key_from_file 'BasicInput.txt'
wait <delay in µsec>	Waiting for a delay in µseconds. Please note that these are emulated µseconds, not a real duration. If your emulator emulates 19968 µsec in actual 10 µsec, it's the 19968 µsec that counts.
	Example : wait 1300455
wait_driveonoff <num>	Wait for drive motor to be started and turned off <num> times. If <num> is not specified, it is 1 by default. (motor on : out &fa7e,1 / motor off:out &fa7e,0)
	Example : wait_driveonoff
wait_vsyncoffon	Wait for vsync to switch from off to on. (on &f5 port, io goes from 0 to 1). If the vsync was already active when the instruction is processed, the emulator must

	wait for the vsync to go off, then wait for the 1st µsec or the vsync goes back to on.
	Example : wait_vsyncoffon
screenshot_name <'name without extension'>	Specifies the name of the next screenshot that will take place: • With the 'screenshot' instruction • With SSM code **#ED #FE** of emulated code
	Example : screenshot_name 'screen01'
screenshot_dir 'screenshot directory'	Specifies the name of the directory where the screenshots are stored. If screenshot_dir is not used, the current emulator directory for screenshots is used.
	Example : screenshot_dir 'c:\SHAKER25\TST\CRTC2\'
screenshot <vsync> screenshot	Generates a screenshot named with screenshot_name instruction, otherwise with the standard name. If the vsync option is specified, the screenshot takes place as soon as the Vsync changes from inactive to active status.
	Example : screenshot
snapshot_name <'name without extension'>	Specifies the name of the next snapshot that will take place: • With the 'snapshot' instruction • With SSM code **#ED #FF** of emulated code
	Example : snapshot_name 'snapshot01'
snapshot <vsync> snapshot	Generates a snapshot named with snapshot_name instruction, otherwise with the standard name. If the vsync option is specified, the snapshot takes place as soon as the Vsync changes from inactive to active status.
	Example : snapshot
csl_load <'name of csl file'>	Load and run a CSL file.
	Example : csl 'SHAKE25B'

The semicolon is used to put comments.
Everything behind a semicolon on a line is ignored.

Note on SSM-CSL management
If the instruction **screenshot_name** is set, then if the emulator executes an **ED FE** Z80A instruction, a screenshot is saved with the name defined with screenshot_name
If the instruction **snapshot_name** is set, then if emulator executes **ED FF** Z80A instruction, a snapshot is saved with the name defined with snapshot_name

Non regression tests
The CSL and SSM standards allows to quickly build a directory of reference images.
These files can then be compared with a file comparison script with the images produced by an evolution of the emulator code, in order to quickly detect any regression.

Several CSL files are associated with SHAKER to allow automatic entry into all tests, in order to automatically generate all SCREENSHOTs.

For further:
An emulator can potentially have an option to record user actions in CSL format.
From the perspective of web distribution, this can avoid creating large videos and automate action sequences for certain games.

Annex: Specific key coding

Key	Sequence
ESC	\(ESC)
TAB	\(TAB)
CAPS LOCK	\(CAP)
SHIFT	\(SHI)
CTRL	\(CTR)
COPY	\(COP)
CLR	\(CLR)
DEL	\(DEL)
RETURN	\(RET)
ENTER	\(ENT)
◀	\(ARL)
▶	\(ARR)
▲	\(ARU)
▼	\(ARD)
F0..F9	\(FN0)..\(FN9)
{	\({)
}	\(})
\	\(\)
‘	\(‘)
No delay next key	\(KOF)