

LOGICAL Help

Winlink2 Command Line Mode (WCL)

The Command Line feature allows Winlink2/Gangpro-8XP to be controlled via command line arguments. Winlink Command Line (WCL) is part of Winlink2 and is automatically enabled when Winlink2 is executed with command line arguments.

This feature is useful for embedding the programmer into another program or batch file. Arguments can be supplied directly on the command line or inserted into a text file.

Winlink2 writes to two status files that provide ready/busy indication and the results of programmer operations. A host program can utilize these files to interface with WCL.

Example batch files and a simple utility to exercise the command line mode are installed into the Winlink2\cmdline folder.

The utility (wlcUtil.exe) is useful for examining the status of programmer operations in command line mode.

In the remainder of this document WCL refers to calling Winlink2.exe with command line arguments.

Hardware Requirements

The command line mode was added to Winlink2 beginning with V1.30. You can check Winlink2's version by running Winlink2 and clicking 'About'. If you are reading this from within Winlink, you already have V1.30+. If you need to upgrade Winlink2, install Winlink2 from the CD-ROM or download from the [Logical Website](#).

The command line mode was added to the Gangpro-8XP beginning with kernel V1.30M. You can check your '8XP's version by running Winlink2 and connecting the '8XP. On startup you will see...

```
Modular Desktop Gang Programmer Vx.xxM  
Resident Algo= 'name of algo' Vx.xx
```

If the Modular Desktop Gang Programmer Vx.xxM is less than V1.30, you will need to update the kernel base.

Click [here](#) for Base Update instructions.

You also should check the Resident Algo version. It should be > V1.20 although earlier versions will work.

Click [here](#) for Module Update instructions.

WCL Environment

By default, Winlink2 is installed into c:\Program Files\Logical Devices\Winlink2. You may want add the path to the Winlink2 installation for convenience but this is not necessary and Winlink2 can locate it's internal files without adding the path.

Note that if you do choose to add the Winlink2 path to autoexec.bat, it must be in the DOS filename format like this.... `PATH= %PATH%;c:\Progra~ 1\Logica~ 1\Winlink2`

For simplicity, Winlink2 Command Line (WCL) uses 2 files to interface with the outside world. STATUS.WCL contains the ready/busy status of WCL. The name of this file cannot be changed. RESULT.WCL contains the result of programmer operations. The name of this result file can be changed with a WCL argument so that a host program can log and store results of operations in different files.

By default, both these files will reside in the CURRENT DIRECTORY in effect at the time WCL is called. The path of both these files can be changed from the default with a WCL argument.

WCL Input Syntax

The syntax is basically....

```
<path> Winlink2<.exe> <@ filename> <cmd1> <arg to cmd1> <cmd2> <arg to cmd2>  
<cmdn> <arg to cmdn>
```

There must be at least one command for Winlink2 to enter Command Line Mode.

Some commands must have an argument.

Command arguments (paths and filenames) must be surrounded by quotes(" ") if the argument contains spaces.

```
Winlink2 -f "c:\My Documents\test files\test.hex" load program verify
```

Windows long pathnames or filenames can be specified with DOS equivalents. (i.e. Program Files -> Progra~1).

Commands and command arguments MUST be separated by spaces.

Commands are NOT case sensitive.

Each command has a 1 or 2 char abbreviation as an alternate.

Command Files

Commands may be placed into a standard text file created with Notepad, Wordpad or your favorite text editor. Enter only one command or command argument per line.

Comments must be on separate lines with '#' as the 1st character of the comment line.

Example cmds.txt:

```
#TEST COMMAND FILE (this is a comment )  
OUTPATH  
c:\my_out_path\  
FILE  
c:\files\test.hex  
LOAD  
BLANK  
PROGRAM  
VERIFY
```

In this case you would call WCL like this... <path>Winlink2 @ <path>cmds.txt

WCL would

- 1) change the output file path to c:\my_out_path\
2) set the Data Filename to c:\files\test.hex
3) Load c:\files\test.hex into the editor.
4) Blank Check the target device(s)
5) Program the target device(s) with the editor data
6) Verify the target device(s) against the editor data.
7) generate c:\my_out_path\result.wcl containing the final result of all the programmer commands.

If cmds.txt is located in a different folder, then you must supply an absolute path or a path relative to the current directory .

Note that by default if you supply the path to the WCL executable when invoking it or put the executable path in autoexec.bat, then the STATUS.WCL,RESULT.WCL, and cmds.txt can all be in one working folder (the current folder when WCL is executed)

What WCL Does Not Control

Currently, WCL does not allow the following....

Module Selection where multiple modules are installed (WCL uses the current module).

Device Selection (this must be performed manually. WCL uses the current selection).

Device Options (these must be setup manually. WCL uses the current options).

Program Options (WCL ignores program options. COPY/PROGRAM will not pre-erase,pre-blank or post-compare/verify).

Winlink Preferences (WCL uses the current Preferences).

Command Reference

(Equivalent shortcuts are enclosed in parentheses.)

Programmer Commands

- ERASE (E)
Erase the TARGET device(s) in the programmer.
- BLANK (B)

Blank Check the TARGET device(s) in the programmer.

- PROGRAM (P)
Program the TARGET device(s) in the programmer with Editor data.
- VERIFY (V)
Verify the TARGET device(s) in the programmer against Editor data.
- COPY (C)
Program the MASTER device data into the TARGET device(s).
- COMPARE (M)
Compare the the TARGET device(s) against the MASTER device.
- CHKSUM (K)
Checksum the MASTER or 1st TARGET device.
- SECURE (U)
Secure the TARGET device(s) in the programmer.
- READ (R)
READ the MASTER or 1st TARGET device into the Editor.

Source Device Commands

- MAST
If both a MASTER and TARGET device(s) are inserted, use the MASTER device in subsequent CHKSUM or READ commands.
If only one or the other (MASTER or TARGETs) is inserted then that device is used.
- TARG (default)
If both a MASTER and TARGET device(s) are inserted, use the 1st TARGET device in subsequent CHKSUM or READ commands.
If only one or the other (MASTER or TARGETs) is inserted then that device is used.

Data File Commands

- FILE (F) [Path\Filename]
Set the Filename for subsequent FILE LOAD or FILE SAVE commands.
Filename must be set before LOAD or SAVE.
Path must be an absolute path or relative to the Winlink2 executable path.
If the Path\Filename contains spaces, then enclose it in quotes(" ");
- SAVE (S)
Save the Editor to Filename.
- LOAD (L)
Load the Editor from Filename

Data File Format Commands

- INTEL (default)
Set the SAVE file format to Intel Hex.
- MOTO
Set the SAVE file format to Motorola Hex.
- TEK

Set the SAVE file format to Tektronix Hex.

- BIN
Set the SAVE file format to Binary.
Note that for LOAD, the file format is autodetected.

Path/File Related Commands

- OUTPATH (OP) [Pathname]
Set the path for RESULT.WCL and STATUS.WCL.
If not set, the PATH in effect when WCL is executed will be used to locate STATUS.WCL.
You must include the trailing backslash '\'.
If the Pathname contains spaces, then enclose it in quotes(" ");
- RESULT (RF) [Filename] or [Pathname\Filename]
Set a new filename for the result file. If the new filename includes a path ('\' char), it overrides the OUTPATH setting for the result file. If the new filename does not include a path , the new filename is appended to OUTPATH to determine the result file location.
If the Pathname contains spaces, then enclose it in quotes(" ");
RESULT only applies to result.wcl and does NOT affect the location of status.wcl.

example:

```
OUTPATH
c:\my_outpath\
RESULT
c:\my_result\new_result.wcl
```

here the status file would be c:\my_outpath\status.wcl and the result file
c:\my_result\new_result.wcl

example:

```
OUTPATH
c:\my_outpath\
RESULT
new_result.wcl
```

here the status file would be c:\my_outpath\status.wcl and the result file
c:\my_outpath\new_result.wcl

Misc Commands

- DEBUG (D)
Do not close WCL after all commands have executed. Remain open and enter normal manual operation mode. WCL will remain open so you can see what occurred. The commands that executed are shown in the Info window on the Winlink2 front panel.

WCL uses STATUS.WCL to indicate it's busy status.

When Winlink2 enters Command Line Mode, it writes the string 'BUSY' to STATUS.WCL. When WCL exits, it it writes the string 'READY' to STATUS.WCL.

A host program can call WCL and then read STATUS.WCL to determine when WCL has finished executing commands.

WCL uses RESULT.WCL to indicate the result of programmer operations. RESULT.WCL is a standard text file. It always contains 10 lines.

Here are some examples of RESULT.WCL files for various programmer commands.

CMD	BLANK	COPY	COMPARE	PROGRAM	VERIFY	CHKSUM	READ	ERASE
MAST	-	M	M	-	-	-	M	-
TARG#1	F	-	-	-	-	-	-	X
TARG#2	P	P	P	P	F	0001FEE2	-	X
TARG#3	P	P	-	P	P	-	-	X
TARG#4	P	-	-	P	-	-	-	X
TARG#5	F	F	F	P	P	-	-	X
TARG#6	-	-	-	-	-	-	-	X
TARG#7	F	-	F	-	-	-	-	X
TARG#8	F	-	-	F	-	-	-	X

Note that unknown commands, file format commands or path commands will not generate a result file. Only programmer commands generate the result file.

Result File Format

Line 1 always denotes the command that generated the result.

Possible values are...

- ERASE
- BLANK
- PROGRAM
- VERIFY
- COPY
- COMPARE
- SECURE
- CHKSUM
- READ

LOAD
SAVE

Line 2 denotes the MASTER device.

Possible values are...

'-' The MASTER device did not participate in the command.

'M' The MASTER device was used as a data source in the command.

'X' THE MASTER device was required by the command but not present or inserted incorrectly.

'NNNNNNNN' is an 8-digit hex number representing the CHKSUM of the MASTER device.

Lines 3 to 10 denote the TARGET device(s). Each line represents one target socket starting with target #1.

Possible values are...

'-' The TARGET device did not participate in the command.

'P' The TARGET device passed the command requirements.

'F' The TARGET device failed the command requirements.

'X' The TARGET device was required by the command but not present or inserted incorrectly.

'NNNNNNNN' is an 8-digit hex number representing the CHKSUM of the TARGET device.

Chained Results

If WCL is executed with more than one command that generates a result, only the final command result is contained in RESULT.WCF. The result of each individual command is chained to the next and a device must pass ALL commands in the chain to be PASSED in the final result. If a device fails ANY command in the chain, it is FAILED in the final result.

Logging Results

If you desire to see the individual command results, then execute WCL with only one command at a time. You can change the name of each result.wcf file by using the RESULT command. This allows a host program to log the results of each operation.

Embedding WCL in other programs

How to call WCL

The Windows API provides several methods for spawning (calling) other programs from inside an application. These include ShellExecute, ShellExecuteEx and CreateProcess. Each varies in complexity and functionality. The simplest ShellExecute is discussed here.

Here is the format of the call (for more details see Help for your compiler or the Windows SDK).

```
HINSTANCE ShellExecute(  
    HWND hwnd, // handle to parent window  
    LPCTSTR lpOperation, // pointer to string that specifies operation to perform  
    LPCTSTR lpFile, // pointer to filename or folder name string  
    LPCTSTR lpParameters, // pointer to string that specifies executable-file parameters  
    LPCTSTR lpDirectory, // pointer to string that specifies default directory  
    INT nShowCmd // whether file is shown when opened  
);
```

Your compiler probably encapsulates ShellExecute() in some form.

Here is the Delphi version from wclUtil.

```
ShellAPI.ShellExecute(0, 'open', pchar(exepath+ 'winlink2.exe'), pchar(edit1.text), nil,  
SW_SHOWNORMAL);
```

here...

exepath is just a string with the path to the Winlink2 installation

edit1.text is a string containing the commands.

pchar() converts Delphi to 'C' strings.

Here is a MS link for Visual Basic...

http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/dnarvb4/html/msdn_shelexec.asp

Here is a Delphi link....

<http://delphi.about.com/od/windowshellapi/l/aa082499.htm>

Here is a VC++ link....

<http://www.codeproject.com/system/newbiespawn.asp>

How to detect when WCL has terminated

You must consider the fact that your Windows program will not wait for WCL to terminate after it is called with ShellExecute. Reliably determining when a spawned program terminates can

get complicated and is beyond the scope of this document. This is the primary reason I included the STATUS.WCL file method to determine when WCL has terminated . It is certainly inelegant but relatively simple.

If you are proficient with the Windows API and want to detect WCL termination via the Windows API, I suggest you see these links for more info

<http://www.festra.com/eng/mtut01.htm>.

<http://www.codeproject.com/system/newbiespawn.asp>.

You can also Google 'shellexecute termination'.

Using STATUS.WCL

Let us examine what you need to do to use STATUS.WCL to detect if WCL is running.

Here is pseudocode...

```
ShellExecute (...); //Start WCL with some command(s)
while (!ready) //wait for WCL to terminate
{
    wait (10-50 ms); //give WCL a chance to open STATUS.WCL
    opened = fopen(STATUS.WCL); //may fail if WCL is writing to STATUS.WCL,
    if (opened) //handle error gracefully
    {
        status_str = fread(STATUS.WCL); //read the 1st line of STATUS.WCL
        fclose(STATUS.WCL); //close STATUS.WCL
        ready = strcmp(status_str,"READY"); //ready is true if 1st line = "READY"
    }
}
continue:
    now open and read RESULT.WCL
```

Essentially, you spawn WCL, then wait in a loop until STATUS.WCL's first line is 'READY'. You may want to limit the loop iterations in case something goes wrong. If your code has STATUS.WCL open when WCL attempts to write to it, WCL will retry 10K times until giving up with an error. Never write to STATUS.WCL

Reading RESULT.WCL

Once you have determined that WCL has terminated, you can then open and parse RESULT.WCL. Do not try to open RESULT.WCL until WCL has exited (STATUS.WCL contains 'READY'). Never write to RESULT.WCL

Minimizing the WCL panel

When embedding WCL, it is not possible to completely eliminate the panel display. You can however, minimize it down to a narrow bar. Winlink will also load faster using this view.

Run Winlink in normal mode, click 'View' on the MAIN MENU and click 'Minimal'.

Using WCLUtil

WclUtil is a simple utility that may help you to embed WCL in your program. It is in the cmdline folder under the Winlink2 Installation folder.

It essentially implements the STATUS.WCL loop described above to provide an easy method to exercise WCL with different arguments and examine the results.

Just click START to start the loop running.

wclUtil expects STATUS.WCL and RESULT.WCL to reside in the folder wclUtil runs from and Winlink2 to reside in the parent folder.

wclUtil waits for a BUSY to READY STATUS.WCL transition and then displays the contents of RESULT.WCL. You can type WCL commands into the edit control at the bottom and click RUN to execute WCL. You can also execute the example .bat files

It also has a reset button in case you hang up WCL somehow with STATUS.WCL still open and need to close it without a reboot.

Several batch files are also provided as examples. You can execute these while wclUtil is running and look at the results.

Be careful when running the batch files with inserted OTP devices. Some of the batch files load a test file and then program the target devices. If you cannot erase your devices, you may render them unusable by programming them with test data.