4D LABS SEMICONDUCTORS

ScriptC

Scripting Utility

REFERENCE MANUAL

Document Revision: 2.1

Document Date: 4th September 2020

ScriptC – Reference Manual

Table of Contents

1. Introduction

ScriptC is a Windows utility designed to run 4D scripts. Using this application allows the automation of tasks that could, otherwise, be executed manually and rigorously. The utility could be used to execute any of the following tasks:

- Load a PmmC to a 4D Labs processor
- Load a program to a 4D Labs processor
- Load a display driver to a 4D Labs processor
- Copy a file from the PC to a uSD card in RAW format
- Copy a file from a RAW uSD card to the PC

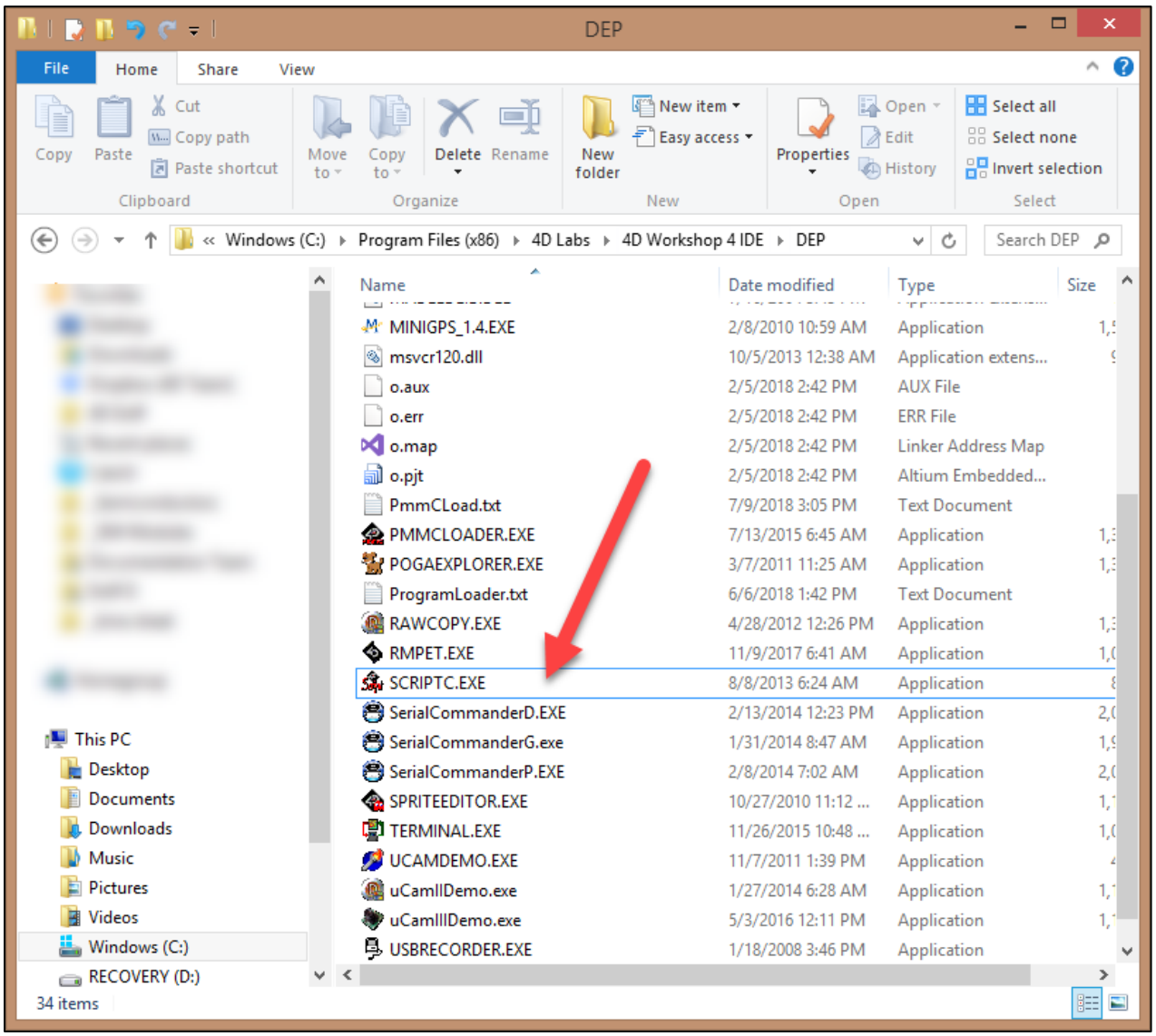
This manual describes the basic commands that can be used in writing a 4D script and the basic options available when running a script with the ScriptC utility.

1.1. Setup

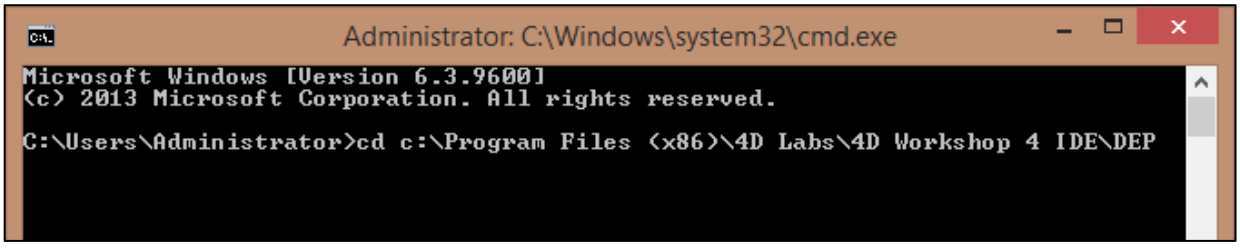
1.1.1. Run ScriptC from the Default Location

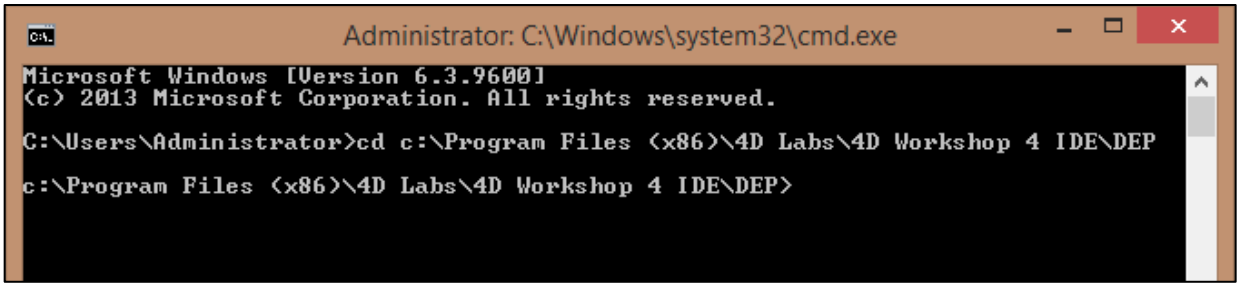
After installing Workshop4, the ScriptC utility (SCRIPTC.EXE) will be, by default, located in the folder:

C:\Program Files (x86)\4D Labs\4D Workshop 4 IDE\DEP

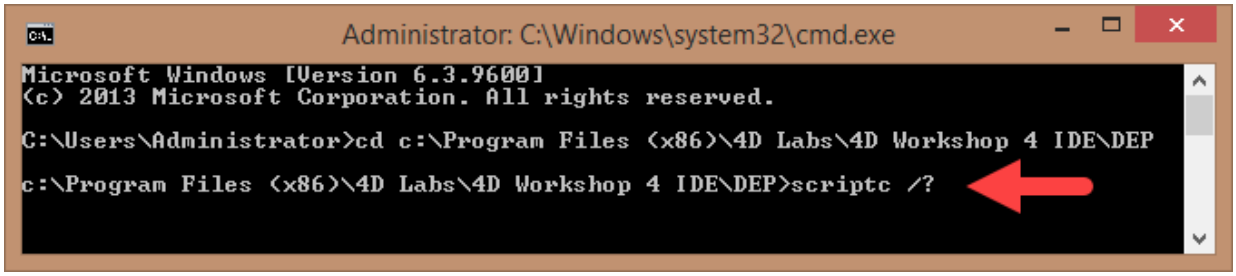


In Windows, open the command line interface and navigate to the above location by using the change directory command, as shown below.

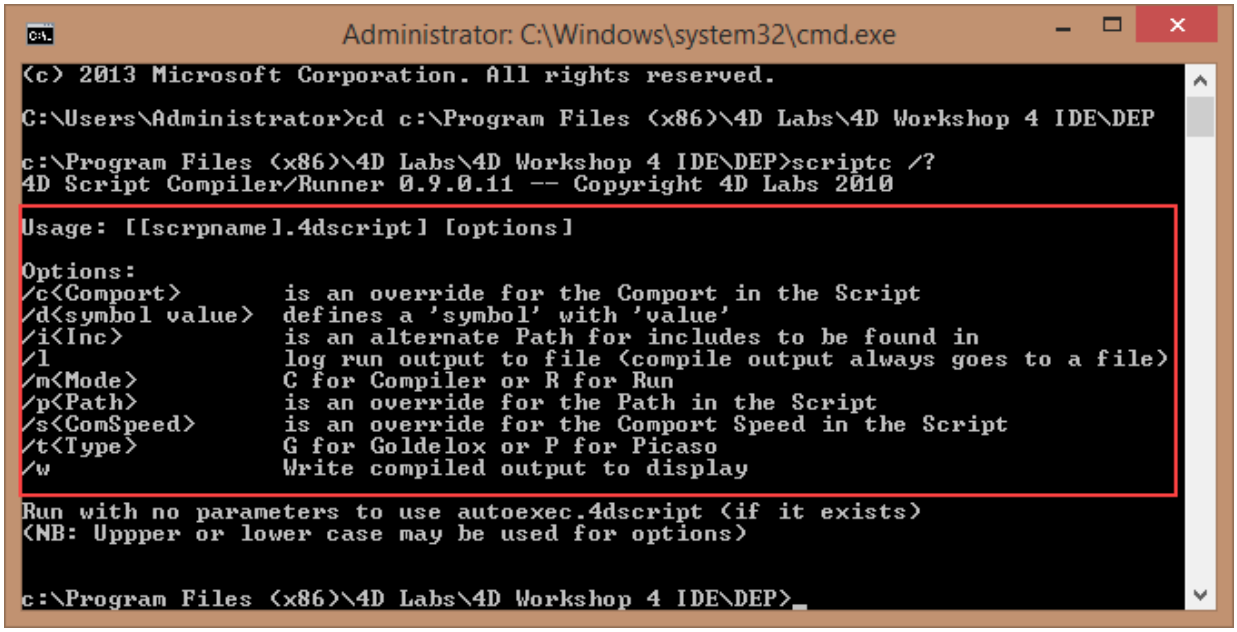




Check the correct usage syntax and options available for ScriptC by typing “scriptc /?”, as shown below.



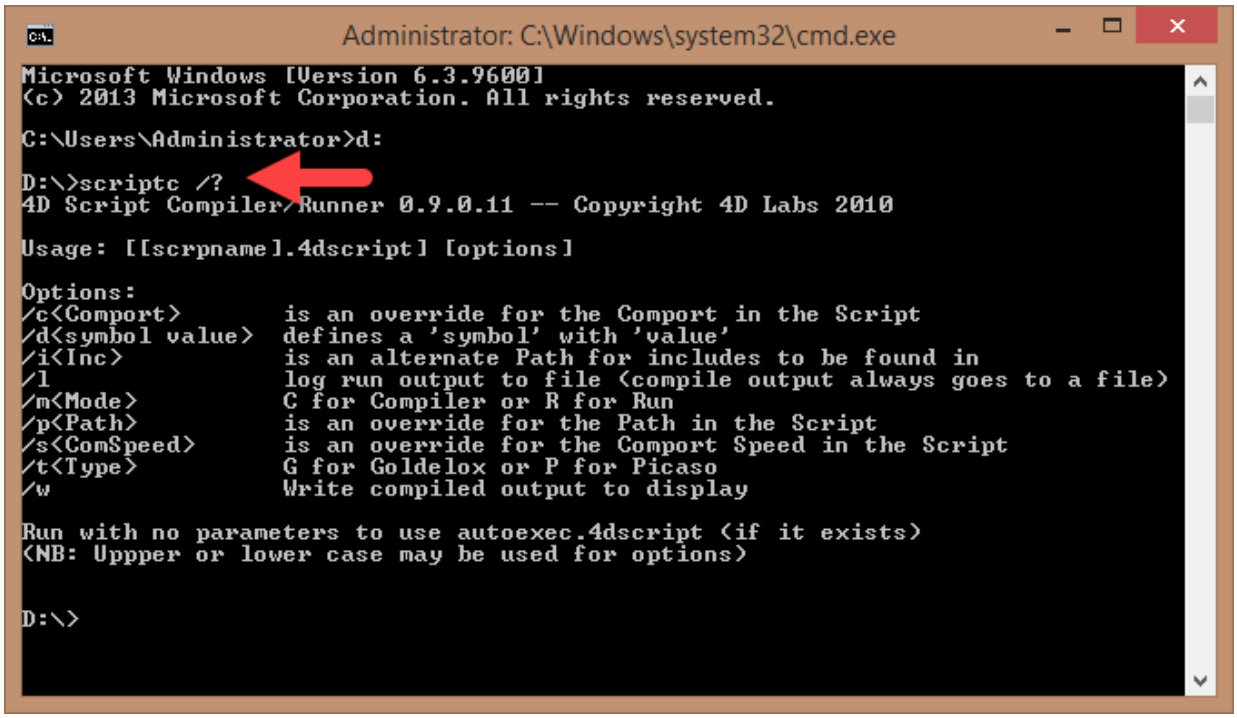
The usage and various options for SriptC should now display.



For more information on the various options available for ScriptC, refer to the section ScriptC Options.

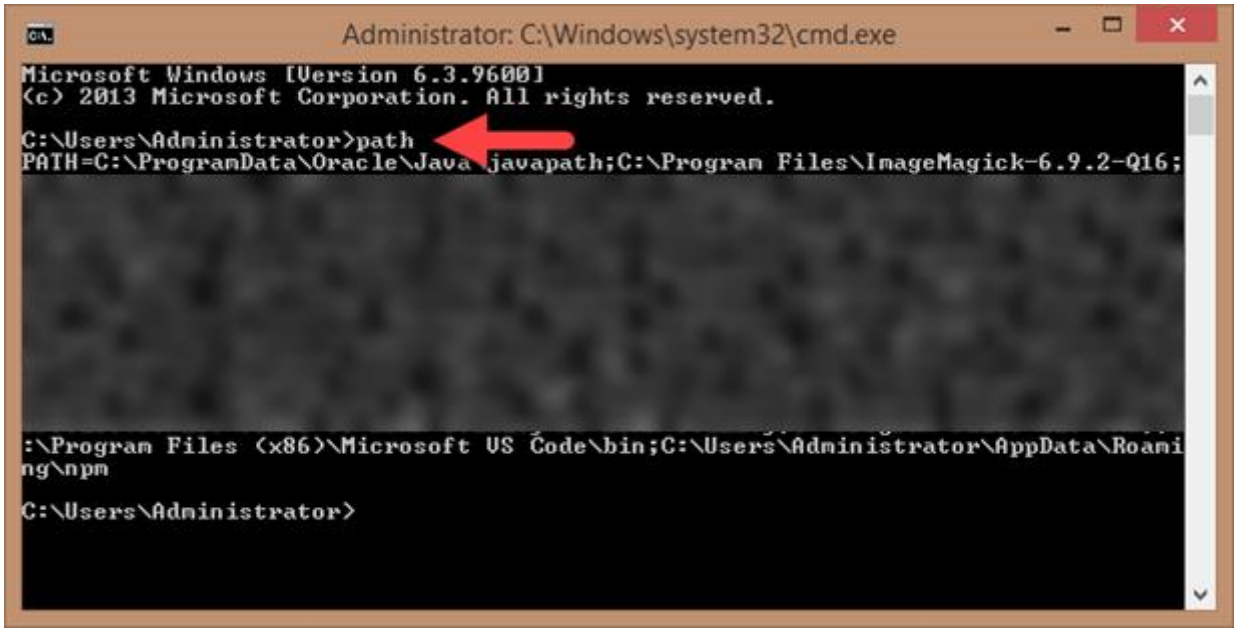
1.1.2. Run ScriptC from another Location

Running the ScriptC utility in the command line interface could be inconvenient as one would need to navigate to the Workshop 4 IDE folder first. A more convenient way is to copy the ScriptC utility to a more accessible location and run it from there. In the example below, ScriptC was first copied to the root of drive D and then it was executed from that location.



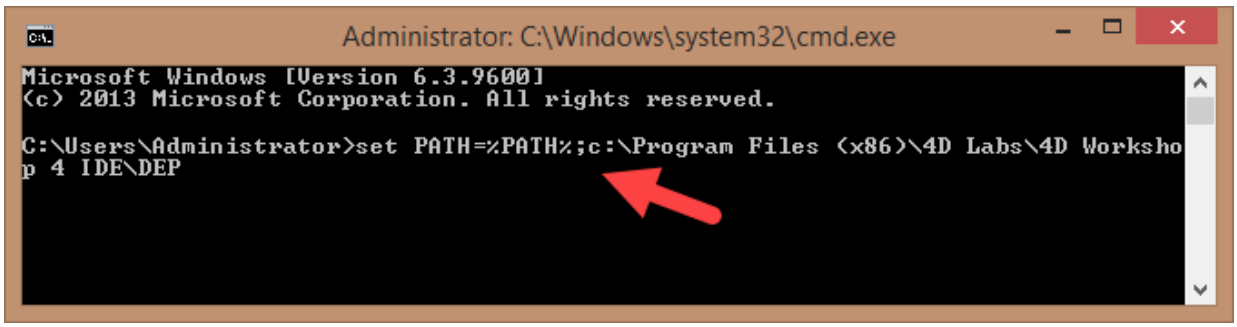
1.1.3. Use the PATH Environment Variable

The PATH environment variable makes it easy to execute commonly used utilities located in their own folders. The PATH environment variable can be edited to include the Workshop4 DEP folder so that the system can easily find the ScriptC utility, regardless of the current directory. The current value of PATH can be checked by typing “path” in the command line interface, as shown below.

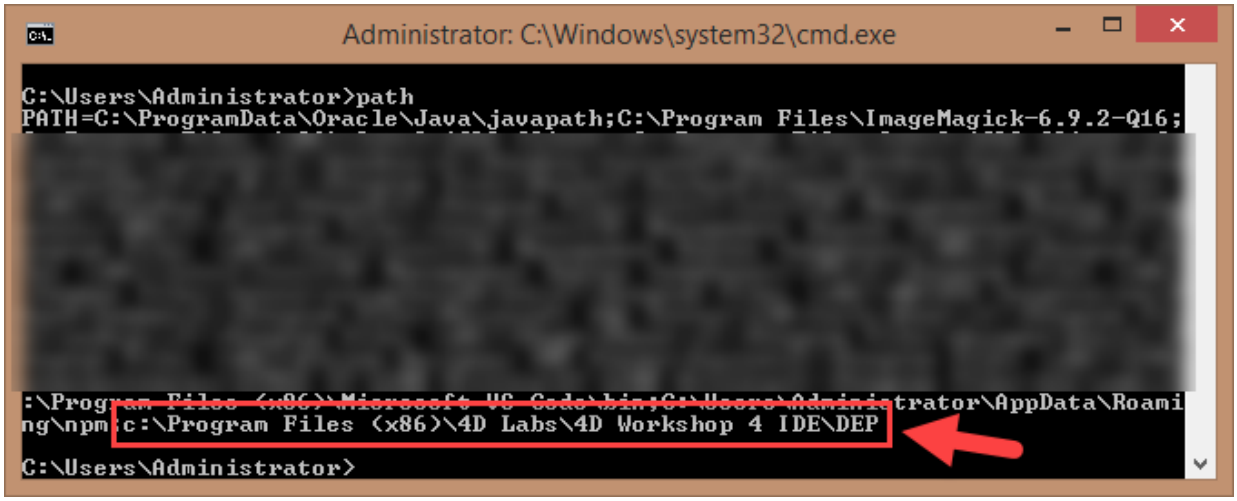


To add the Workshop4 DEP folder to PATH, type the command shown below.

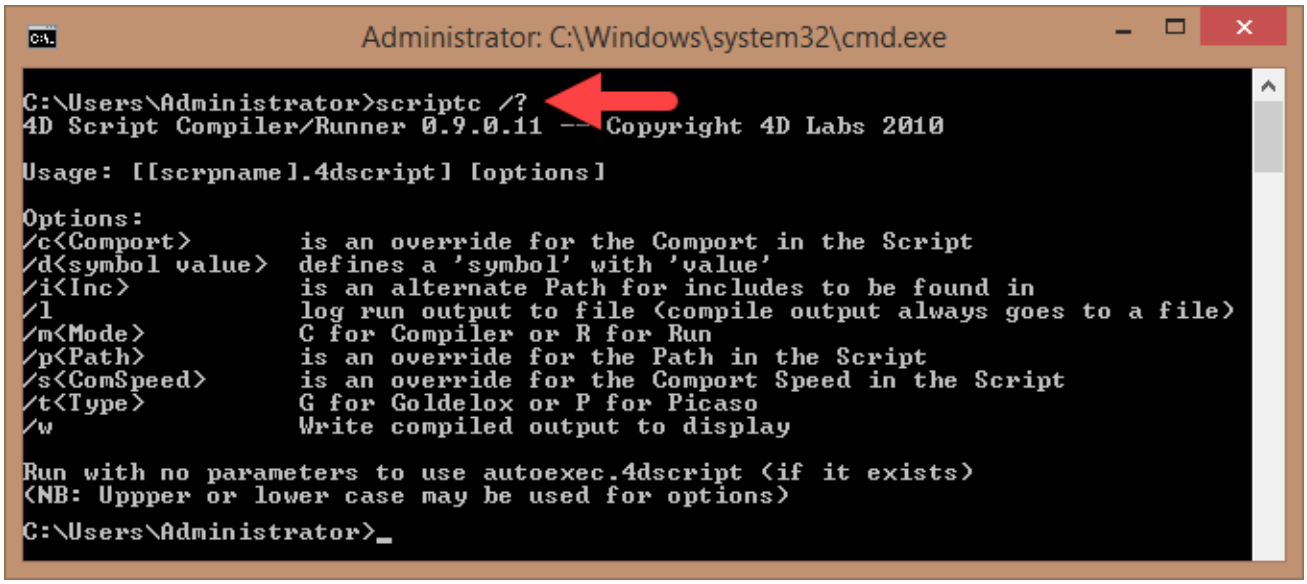
```
set PATH=%PATH%;c:\Program Files (x86)\4D Labs\4D Workshop 4 IDE\DEP
```



Now check the value of PATH again.



The ScriptC utility can now be found by the system regardless of the current directory.



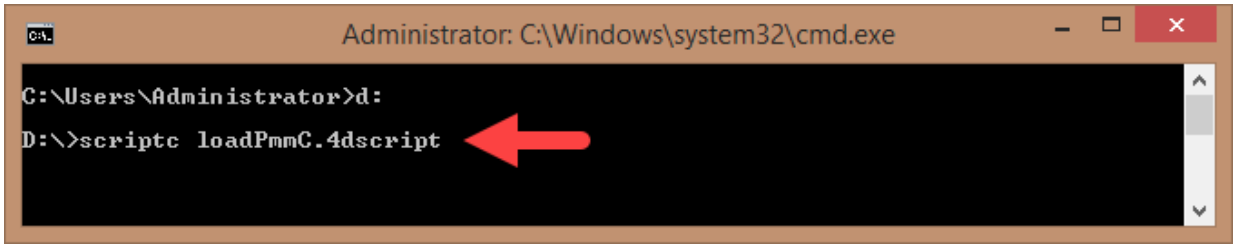
Note: The above procedure for editing the PATH environment variable is valid for the current console session only.

1.2. Execution

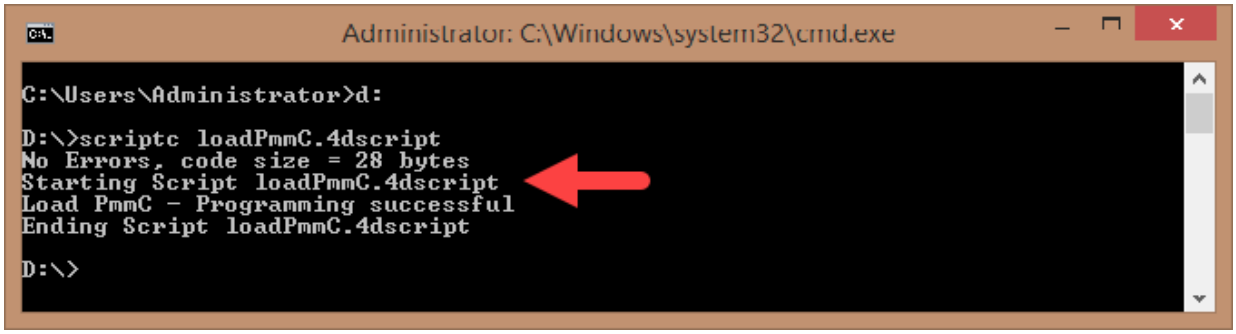
1.2.1. Run a 4D Script File with ScriptC

To run a 4D script file with the ScriptC utility using the command line interface, navigate to the location of the script file and type the command as shown below.

```
scriptc <script filename here>
```



ScriptC should now execute the script and print the result to the console window.



In the example shown above, the 4D script file "loadPmmC.4Dscript" is assumed to be located in the root of drive D.

Note: Make sure that ScriptC and the console window session have been setup as described in any of the options provided in the section Setup.

Note: Sample scripts for accomplishing various tasks can be found in the appendix sections.

Note: Additional sample scripts can be found in the Workshop4 folder: C:\Users\Public\Documents\4D Labs\SCRIPTS

1.2.2. Options

ScriptC has various options that could be useful when running 4D scripts. For more information, refer to the section ScriptC Options.

1.2.3. Automation with Batch Files

Batch files can be used to run 4D scripts with the ScriptC utility. For more information, refer to the appendix section Using a Batch File.

2. Script Commands

A 4D script file can be created and edited using any text editor. 4D script files have the filename extension ".4DScript". The following is a summary of script commands.

Directives

- #define
- #include
- #run

Macros

- $4DGLLoadprogram
- $LoadPmmC
- $writePCuSD
- $readPCuSD

Extended Macros

- $TimeOff
- $TimeOn

2.1. Directives

Directives are lines included in the program but are not program statements. These lines are always preceded by a hash sign (#). They are executed before the actual compilation of code begins.

They extend only across a single line of code. As soon as a newline character is found, the directive is considered to end. No semicolon **";"** is expected at the end of the directive.

Summary of commands in this section:

- #define
- #include
- #run

2.1.1. #define

Syntax	**#define ("Name", "Substitution")**	
Arguments	**"Name", "Substitution"**	
	Name	Source to be substituted
	Substitution	The replacement text or value
Description	This can be used to define replacement for parameters so that they can be set from the command line.	
Example	`#define red 0xf800` `#define usdfile  "c:\usd0.hex"`	

2.1.2. #include

Syntax	**#include("Filename")**	
Arguments	**"Filename"**	
	Filename	Name of the file to be included
Description	This can be used to include other files into the script.	
Example	`#include "4DScript_16bitColours.inc"`	

2.1.3. #run

Syntax	**#run("Platform", "Comport", "Speed", "WrapCol", "WrapTrunc")**	
Arguments	**"Platform", "Comport", "Speed", "WrapCol", "WrapTrunc"**	
	Platform	Picaso or Goldelox (see description below)
	Comport	The comm port to use.
	Speed	The maximum speed of the comm port used during downloads. 9600 is used normally.
	WrapCol	The number of bytes after which wrapping or truncation occurs in the compile listing.
	WrapTrunc	Wrap or Trunc Specifies whether the compile listing is wrapped or truncated when Wrapcol is reached.
Description	Set script run and options. This must be the first line of a script. When using Pixxi-28, Pixxi-44 or Diablo16, use Picaso for the platform.	
Example	`#run(Picaso,COM4,9600,5,Wrap)`	

2.2. Macros

Given below is the detailed command set for macros that are executed from the PC while the display module is connected to it. These commands begin with a **"$"** sign.

Summary of commands in this section:

- $4DGLLoadprogram
- $LoadPmmC
- $writePCuSD
- $readPCuSD
- Extended Macros
  - $TimeOff
  - $TimeOn

2.2.1. $4DGLLoadprogram

Syntax	**$4DGLLoadprogram(“4DGLprogram”, “memory”)**	
Arguments	**“4DGLprogram”, “memory”**	
	4DGLprogram	The filename of a compiled 4DGL program to be loaded onto a display
	memory	**Ram:** Target is RAM on the processor. **Flash:** Target is Flash on the processor.
Response	**acknowledge**	
	acknowledge	**06**(hex) : ACK byte if successful **15**(hex) : NAK byte if unsuccessful
Description	Loads the specified program to the display. The file extension must **not** be specified.	
Example	`$4DGLLoadProgram("c:\gfx2_BigDemo",Ram)`	

2.2.2. $LoadPmmC

Syntax	**$LoadPmmC("PmmCName")**	
Arguments	**"PmmCName"**	
	PmmCName	The filename of the PmmC to be loaded.
Response	**acknowledge**	
	acknowledge	**06**(hex) : ACK byte if successful **15**(hex) : NAK byte if unsuccessful
Description	Loads a PmmC to the processor.	
Example	`$LoadPmmC("c:\PmmC\uLCD-43PT-R44.PmmC")`	

2.2.3. $writePCuSD

Syntax	**$WritePCuSD("pcFile", "uSDDrive", "OverWrite", "startSector")**	
Arguments	**"pcFile", "uSDDrive", "OverWrite", "startSector"**	
	pcFile	The file to read the sectors from.
	uSDDrive	The drive the uSD card is in, eg G:
	OverWrite	OverWrite or Preserve What to do if the uSD card contains a file system. “Preserve” abandons the write.
	startSector	The starting sector to write to.
Description	This writes the file on the PC to a uSD card mounted to the PC starting at *startSector*. If the final sector is not filled it is padded with hex 00s.	
Example	`// overwrite start of uSD card mounted on drive F with file "IMAGEG~1.gci"` `$writePCuSD("IMAGEG~1.gci","F:",OverWrite, 0)`	

2.2.4. $readPCuSD

Syntax	**$readPCuSD("pcFile", "uSDDrive", "StartSector", "#Sectors")**	
Arguments	**"pcFile", "uSDDrive", "StartSector", "#Sectors"**	
	pcFile	The file to write the sectors to.
	uSDDrive	The drive the uSD card is in, eg G:.
	StartSector	The starting sector to read.
	#sectors	The number of sectors to read.
Description	This reads *#sectors* sectors from the uSD card mounted to drive *uSDDrive* of the PC, starting at sector *StartSector* and saves them to a file on the PC with the filename "*pcFile*".	

Example	`//save the first 23 sectors of the uSD card mounted on drive F`
	`//to the file "Sectors.hex"`
	`$readPCuSD("Sectors.hex","F:",0,23)`	

2.2.5. Extended Macros

COMMAND	DESCRIPTION
$TimeOff	This turns off logging of times for each command (default).
$TimeOn	This turns on logging of times for each command.

3. ScriptC Options

This section describes the different options available for the ScriptC utilty. To display the options in the console window, type the command below.

```
scriptc /?
```

```
D:\>scriptc /?
4D Script Compiler/Runner 0.9.0.11 -- Copyright 4D Labs 2010

Usage: [[scrpname].4dscript] [options]

Options:
/c<Comport>       is an override for the Comport in the Script
/d<symbol value>  defines a 'symbol' with 'value'
/i<Inc>           is an alternate Path for includes to be found in
/l                log run output to file (compile output always goes to a file)
/m<Mode>          C for Compiler or R for Run
/p<Path>          is an override for the Path in the Script
/s<ComSpeed>      is an override for the Comport Speed in the Script
/t<Type>          G for Goldelox or P for Picaso
/w                Write compiled output to display

Run with no parameters to use autoexec.4dscript (if it exists)
(NB: Uppper or lower case may be used for options)
```

Summary of options described in this section:

- Comm Port Override
- Define a Symbol
- Comm Speed Override
- Alternate Include Path
- Output Logs

3.1.1. Comm Port Override

Option	/c<Comport>
Description	This option can be used to override the comm port specified in the script.

Example	//override the comm port specified in the script "LoadPmmC.4DScript"
	// with "COM4"
	
	scriptc LoadPmmC.4DScript /ccom4

3.1.2. Define a Symbol

Option	/d<symbol value>
Description	This option can be used to define a symbol *symbol* with the value *value*.

Example	`//define a symbol "pmmcname" with the value "c:\PmmC\Diablo16-R19.PmmC".`
	`scriptc LoadPmmC.4DScript /dpmmcname "c:\PmmC\Diablo16-R19.PmmC"`
	`//LoadPmmC.4DScript could then use the symbol pmmcname.`

3.1.3. Comm Speed Override

Option	/s<ComSpeed>
Description	This option can be used to override the comm port speed specified in the script.

Example	`//override the comm port speed specified in the script "LoadPmmC.4DScript"`
	`//with the value 9600`
	
	`scriptc LoadPmmC.4DScript /s9600`

3.1.4. Alternate Include Path

Option	/i<Inc>
Description	This option can be used to add another path for include files, besides the current directory.
Example	//include the location "d:\inc" scriptc LoadPmmC.4DScript /id:\inc

3.1.5. Output Logs

Option	/l
Description	By default, runtime logs of the ScriptC utility are printed to the console window. When used, "/l" causes the logs to be written to a file in the current directory, instead of the console window.

Example	`//ScriptC creates the file "LoadPmmC.log" in the current directory`
	`//and outputs the runtime logs to it`
	`scriptc LoadPmmC.4DScript /l`

4. Appendix A: Script for Loading a PmmC

4.1. Goldelox

4.1.1. Write the Script

Create the file “LoadPmmC.4DScript” and put the contents below.

```
#Run(Goldelox,COM6,9600,5,Wrap)

/*
4D Script Sample
Note: In this case the Goldelox / Picaso specification is meaningless as there is
no code created that differs between Goldelox and Picaso.
*/

$LoadPmmC("d:\Goldelox\uOLED-96-G2-R26.PmmC")
```

4.1.2. Requirements

4.1.2.1. PmmC File

The script assumes that the PmmC file “uOLED-96-G2-R26.PmmC" is in the folder “d:\Goldelox”. Modify the script and put the correct PmmC filename and path accordingly if needed.

4.1.2.2. Comm Port

The script assumes that the display module is connected to comm port COM6. Connect the display module to the PC. Modify the script and put the correct comm port if needed.

4.1.3. Run the Script

4.1.3.1. Setup

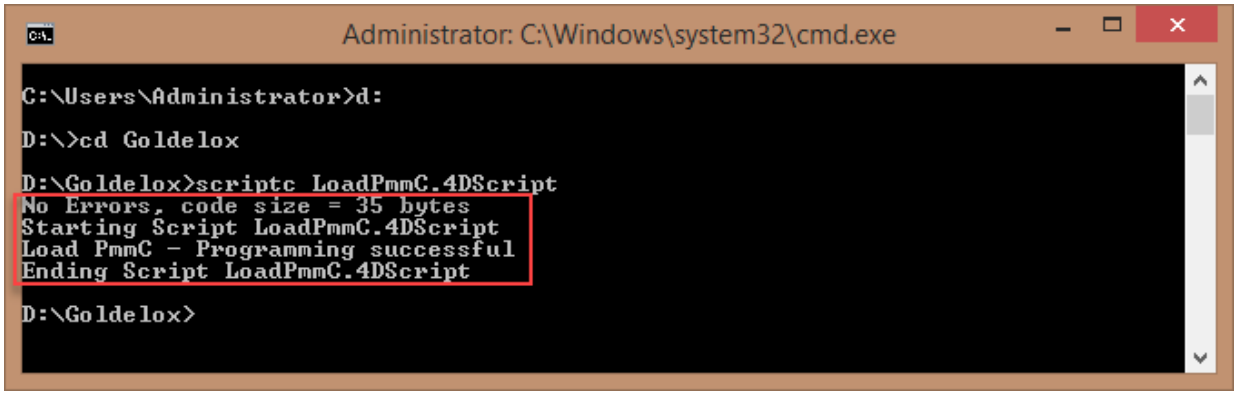
Open and setup the console window as discussed in the section Setup under the section Introduction.

4.1.3.2. Execution

Run the script by following the steps described in the section Execution under the section Introduction.

4.1.3.3. Output

ScriptC should now print the results to the console window.



4.2. Picaso, Pixxi-28 and Pixxi-44

Picaso commands are also relevant for Pixxi, so use the Picaso examples when using Pixxi-28 or Pixxi-44 too.

4.2.1. Write the Script

Create the file “LoadPmmC.4DScript” and put the contents below.

```
#Run(Picaso,COM6,9600,5,Wrap)

/*
4D Script Sample
Note: In this case the Goldelox / Picaso specification is meaningless as there is
no code created that differs between Goldelox and Picaso.
*/

$LoadPmmC("d:\Picaso\uLCD-43PCT-R44.PmmC")
```

4.2.2. Requirements

4.2.2.1. PmmC File

The script assumes that the PmmC file “uLCD-43PCT-R44.PmmC" is in the folder “d: \Picaso”. Modify the script and put the correct PmmC filename and path accordingly if needed.

4.2.2.2. Comm Port

The script assumes that the display module is connected to comm port COM6. Connect the display module to the PC. Modify the script and put the correct comm port if needed.

4.2.3. Run the Script

4.2.3.1. Setup

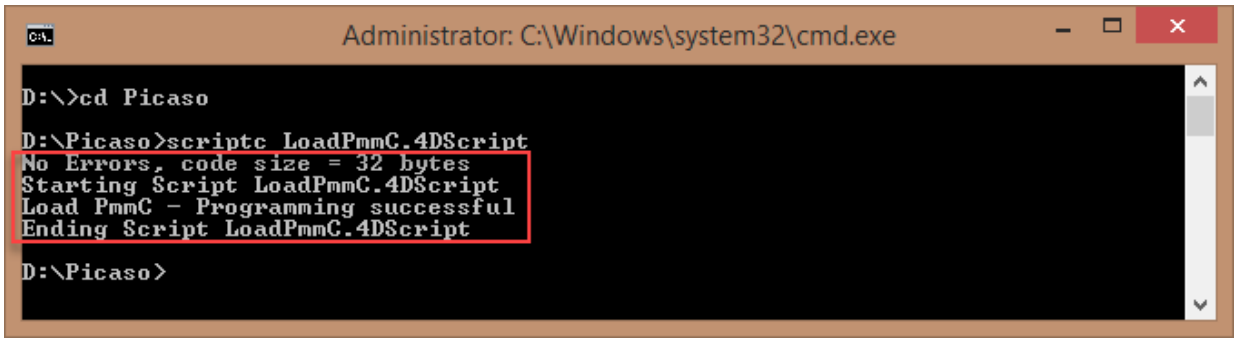
Open and setup the console window as discussed in the section Setup under the section Introduction.

4.2.3.2. Execution

Run the script by following the steps described in the section Execution under the section Introduction.

4.2.3.3. Output

ScriptC should now print the results to the console window.



In the example above, the script “LoadPmmC.4DScript” is located in “D:\Picaso”.

4.3. Diablo16

4.3.1. Write the Script

Create the file “LoadPmmC.4DScript” and put the contents below.

```
#Run(Picaso,COM6,9600,5,Wrap)

/*
4D Script Sample
Note: In this case the Goldelox / Picaso specification is meaningless as there is
no code created that differs between Goldelox and Picaso.
*/

$LoadPmmC("d:\Diablo16\Diablo16-R20.PmmC")
```

4.3.2. Requirements

4.3.2.1. PmmC File

The script assumes that the PmmC file “Diablo16-R20.PmmC" is in the folder “d:\Diablo16”. Modify the script and put the correct PmmC filename and path accordingly if needed.

4.3.2.2. Comm Port

The script assumes that the display module is connected to comm port COM6. Connect the display module to the PC. Modify the script and put the correct comm port if needed.

4.3.3. Run the Script

4.3.3.1. Setup

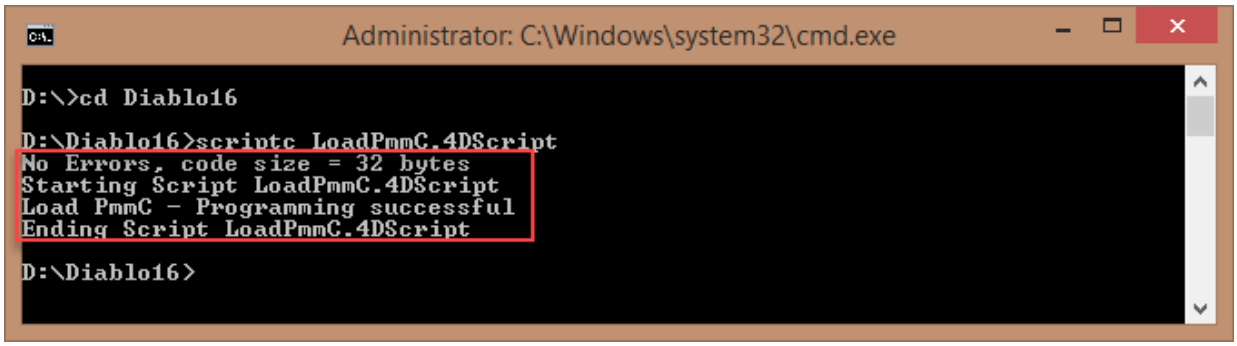
Open and setup the console window as discussed in the section Setup under the section Introduction.

4.3.3.2. Execution

Run the script by following the steps described in the section Execution under the section Introduction.

4.3.3.3. Output

ScriptC should now print the results to the console window.



In the example above, the script “LoadPmmC.4DScript” is located in “D:\Diablo16”.

5. Appendix B: Script for Loading a Driver

5.1.1. Write the Script

Create the file “LoadDriver.4DScript” and put the contents below.

```
#Run(Picaso,COM6,9600,5,Wrap)

/*
4D Script Sample
Note: In this case the Goldelox / Picaso specification is meaningless as there is
no code created that differs between Goldelox and Picaso.
*/

$LoadPmmC("d:\Diablo16\uLCD-43DT-B-D160516.4Drv")
```

5.1.2. Requirements

5.1.2.1. Driver File

The script assumes that the driver file “uLCD-43DT-B-D160516.4Drv" is in the folder “d:\Diablo16”. Modify the script and put the correct driver filename and path accordingly if needed.

5.1.2.2. Comm Port

The script assumes that the display module is connected to comm port COM6. Connect the display module to the PC. Modify the script and put the correct comm port if needed.

5.1.3. Run the Script

5.1.3.1. Setup

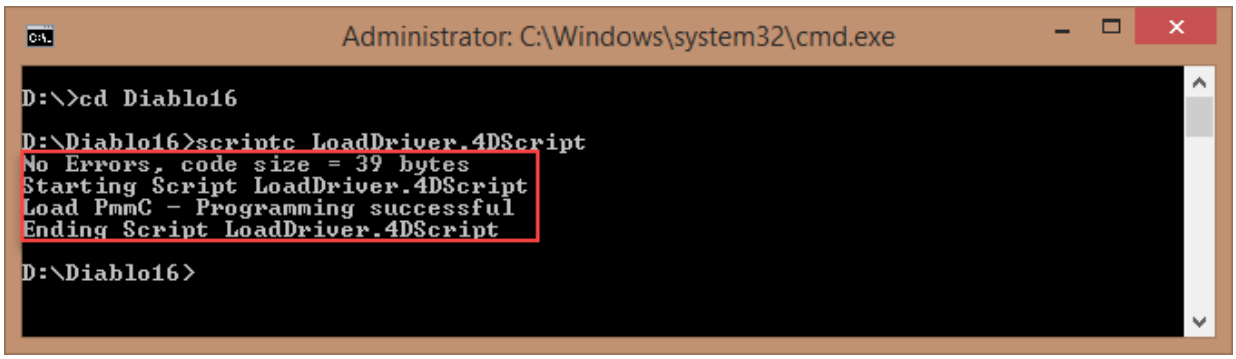
Open and setup the console window as discussed in the section Setup under the section Introduction.

5.1.3.2. Execution

Run the script by following the steps described in the section Execution under the section Introduction.

5.1.3.3. Output

ScriptC should now print the results to the console window.



In the example above, the script “LoadDriver.4DScript” is located in “D:\Diablo16”.

6. Appendix C: Script for Loading a Program

6.1. Goldelox

6.1.1. Write the Script

Create the file “LoadProgram.4DScript” and put the contents below.

```
#Run(Goldelox,COM6,9600,5,Wrap)

/*
4D Script Sample
Note: In this case the Goldelox / Picaso specification is meaningless as there is no code created that differs between Goldelox and Picaso.
Note: The Ram / Flash option is ignored for Goldelox programs
*/

$4DGLLoadProgram("D:\Goldelox\HelloWorld",Ram)
```

6.1.2. Requirements

6.1.2.1. Program Files

The script assumes that the files “HelloWorld.4XE" and “HelloWorld.cfg” are in the folder “d:\Goldelox”. Modify the script and put the correct program filenames and path accordingly if needed.

6.1.2.2. Comm Port

The script assumes that the display module is connected to comm port COM6. Connect the display module to the PC. Modify the script and put the correct comm port if needed.

6.1.3. Run the Script

6.1.3.1. Setup

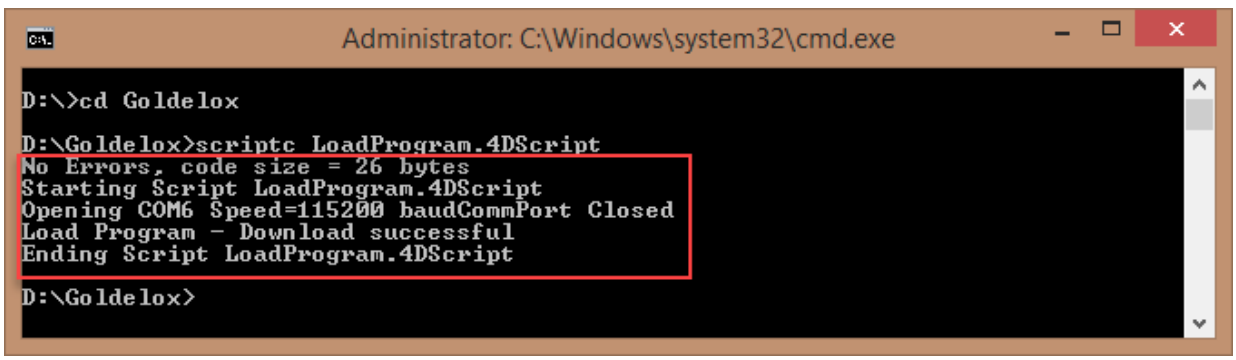
Open and setup the console window as discussed in the section Setup under the section Introduction.

6.1.3.2. Execution

Run the script by following the steps described in the section Execution under the section Introduction.

6.1.3.3. Output

ScriptC should now print the results to the console window.



In the example above, the script “LoadProgram.4DScript” is located in “D:\Goldelox”.

6.2. Picaso, Pixxi-28 and Pixxi-44

6.2.1. Write the Script

Create the file “LoadProgram.4DScript” and put the contents below.

```
#Run(Picaso,COM6,9600,5,Wrap)

/*
4D Script Sample
Note: In this case the Goldelox / Picaso specification is meaningless as there is
no code created that differs between Goldelox and Picaso.
*/

$4DGLLoadProgram("D:\Picaso\HelloWorld",Ram)
```

6.2.2. Requirements

6.2.2.1. Program Files

The script assumes that the files “HelloWorld.4XE" and “HelloWorld.cfg” are in the folder “d:\Picaso”. Modify the script and put the correct program filenames and path accordingly if needed.

6.2.2.2. Comm Port

The script assumes that the display module is connected to comm port COM6. Connect the display module to the PC. Modify the script and put the correct comm port if needed.

6.2.3. Run the Script

6.2.3.1. Setup

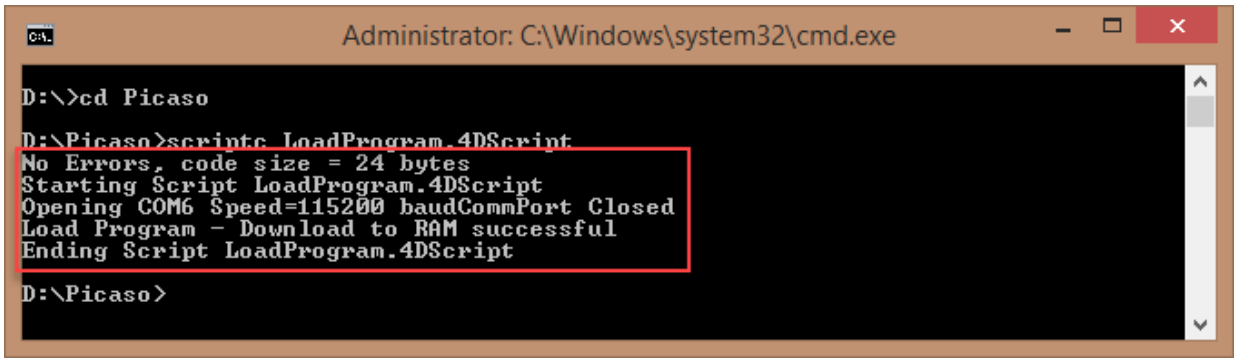
Open and setup the console window as discussed in the section Setup under the section Introduction.

6.2.3.2. Execution

Run the script by following the steps described in the section Execution under the section Introduction.

6.2.3.3. Output

ScriptC should now print the results to the console window.



In the example above, the script “LoadProgram.4DScript” is located in “D:\Picaso”.

6.3. Diablo16

6.3.1. Write the Script

Create the file “LoadProgram.4DScript” and put the contents below.

```
#Run(Picaso,COM6,9600,5,Wrap)

/*
4D Script Sample
Note: In this case the Goldelox / Picaso specification is meaningless as there is
no code created that differs between Goldelox and Picaso.
*/

$4DGLLoadProgram("D:\Diablo16\HelloWorld",Ram)
```

6.3.2. Requirements

6.3.2.1. Program Files

The script assumes that the files “HelloWorld.4XE" and “HelloWorld.cfg” are in the folder “d:\Diablo16”. Modify the script and put the correct program filenames and path accordingly if needed.

6.3.2.2. Comm Port

The script assumes that the display module is connected to comm port COM6. Connect the display module to the PC. Modify the script and put the correct comm port if needed.

6.3.3. Run the Script

6.3.3.1. Setup

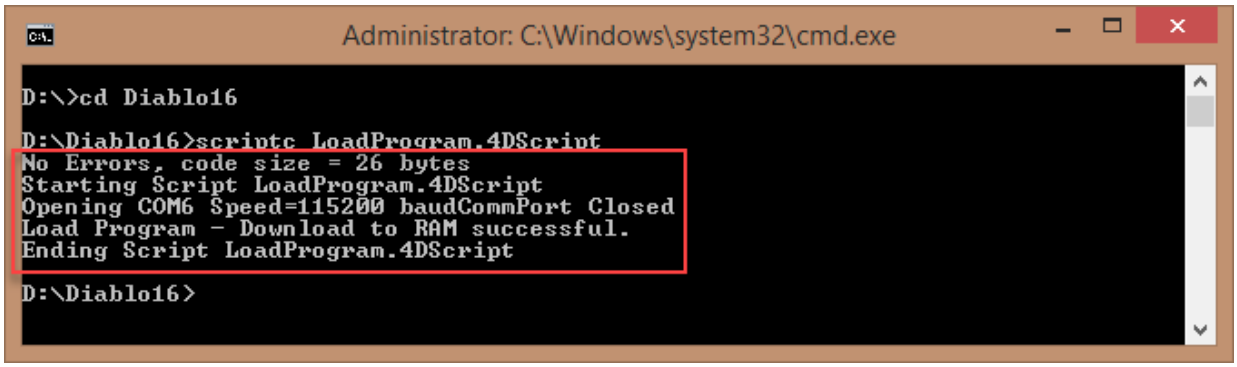
Open and setup the console window as discussed in the section Setup under the section Introduction.

6.3.3.2. Execution

Run the script by following the steps described in the section Execution under the section Introduction.

6.3.3.3. Output

ScriptC should now print the results to the console window.



In the example above, the script “LoadProgram.4DScript” is located in “D:\Diablo16”.

7. Appendix D: Script for Writing a File to the uSD Card

7.1. Write the Script

Create the file “WriteUSD.4DScript” and put the contents below.

```
#Run(Picaso,COM4,256000,5,Wrap)
$writePCuSD("D:\uSD\IMAGEG~1.gci","F:",OverWrite, 0)
```

7.1.1. Source File

The script assumes that the file “IMAGEG~1.gci " is in the folder “D:\uSD”. Modify the script and put the correct source filename and path accordingly if needed.

7.1.2. uSD Card Drive

The script assumes that the target uSD card is mounted on drive F. Mount the uSD card to the PC. Modify the script and put the correct drive if needed.

7.2. Run the Script

7.2.1. Setup

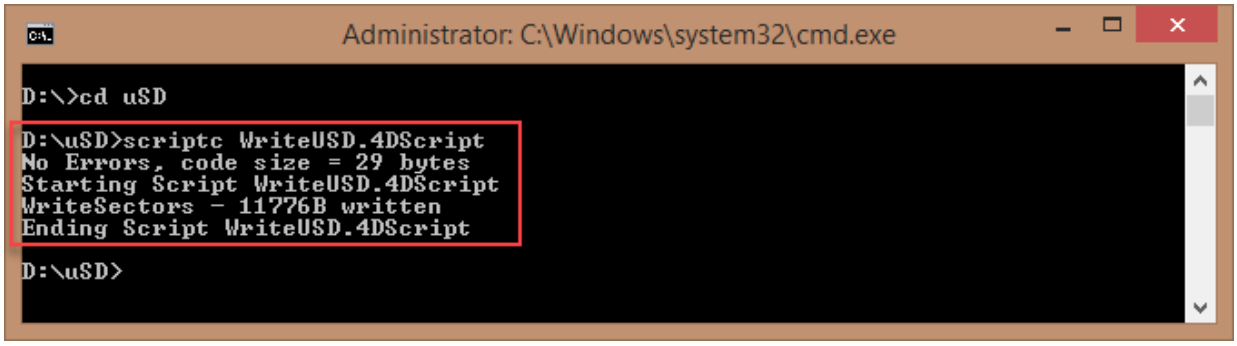
Open and setup the console window as discussed in the section Setup under the section Introduction.

7.2.2. Execution

Run the script by following the steps described in the section Execution under the section Introduction.

7.2.3. Output

ScriptC should now print the results to the console window.



In the example above, the script “WriteUSD.4DScript” is located in “D:\uSD”.

8. Appendix D: Scrtip for Copying a File from the uSD Card

8.1. Write the Script

Create the file “ReadUSD.4DScript” and put the contents below.

```
#Run(Picaso,COM4,256000,5,Wrap)
$readPCuSD("Sectors.hex","F:",0,23)
```

8.1.1. Destination File

ScriptC will create the file "Sectors.hex" in the same folder where the script file is located. Modify the script and put the correct destination filename and path accordingly if needed.

8.1.2. uSD Card Drive

The script assumes that the source uSD card is mounted on drive F. Mount the uSD card to the PC. Modify the script and put the correct drive if needed

8.2. Run the Script

8.2.1. Setup

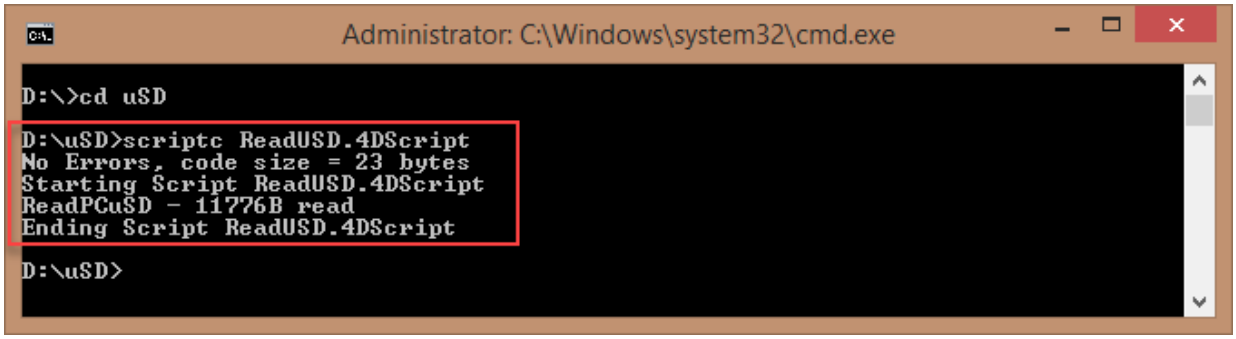
Open and setup the console window as discussed in the section Setup under the section Introduction.

8.2.2. Execution

Run the script by following the steps described in the section Execution under the section Introduction.

8.2.3. Output

ScriptC should now print the results to the console window.



In the example above, the script “ReadUSD.4DScript” is located in “D:\uSD”.

9. Appendix E: Using the Extended Macros

9.1. Write the Script

Create the file “WriteAndReadUSD.4DScript” and put the contents below.

```
#Run(Picaso,COM4,256000,5,Wrap)
$timeon
$writePCuSD("D:\uSD\IMAGEG~1.gci","F:",OverWrite, 0)
$timeoff
$readPCuSD("D:\uSD\Sectors.hex","F:",0,23)
```

9.1.1. Source and Destination Files

The script assumes that the file “IMAGEG~1.gci " is in the folder “D:\uSD”. Modify the script and put the correct source filename and path accordingly if needed.

The script will create the file "Sectors.hex" in the folder “D:\uSD”. Modify the script and put the correct destination filename and path accordingly if needed.

9.1.2. uSD Card Drive

The script assumes that the uSD card is mounted on drive F. Mount the uSD card to the PC. Modify the script and put the correct drive if needed.

9.2. Run the Script

9.2.1. Setup

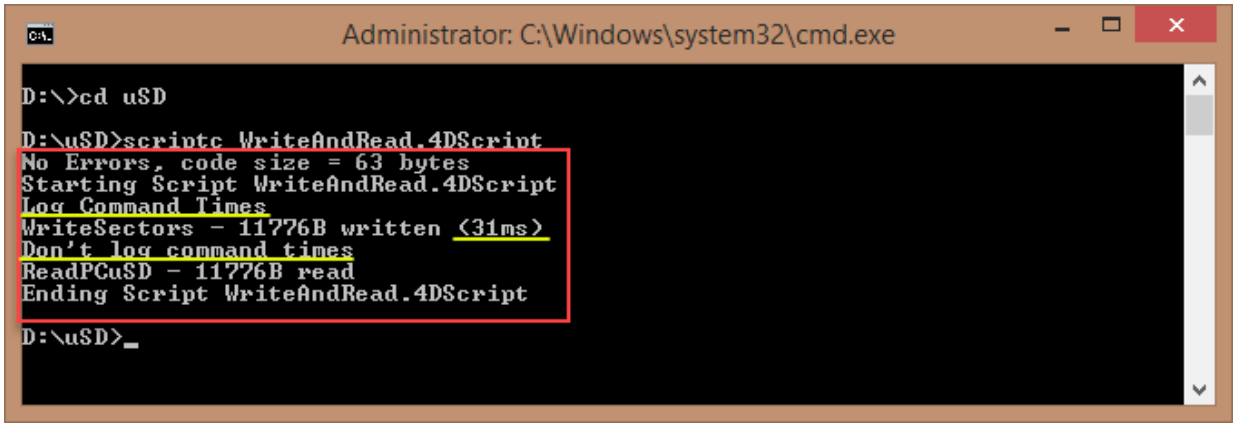
Open and setup the console window as discussed in the section Setup under the section Introduction.

9.2.2. Execution

Run the script by following the steps described in the section Execution under the section Introduction.

9.2.3. Output

ScriptC should now print the results to the console window.



In the example above, the script “WriteAndReadUSD.4DScript” is located in “D:\uSD”. Note also that time logs are enabled for the $writePCuSD() command and disabled for the $readPCuSD() command, due to the use of the extended macros “$timeon” and “$timeoff” in the script.

10. Appendix F: Using a Batch File

Batch files can also be utilized to run 4D script files with ScriptC. Below is an example of this.

10.1. Write the Scripts

Create the file "LoadProgram.4DScript" and put the contents below.

```
#Run(Picaso,COM6,9600,5,Wrap)
/*
4D Script Sample
Note: In this case the Goldelox / Picaso specification is meaningless as there is
no code created that differs between Goldelox and Picaso.
*/

$4DGLLoadProgram(programlocation,Ram)
```

10.1.1. Write the Batch File Script

Create a batch file (for example BatchFileTest.bat) and put the contents below.

```
@ECHO OFF
REM temporarily add default location of ScriptC to PATH so the system can find it
set PATH=%PATH%;c:\Program Files (x86)\4D Labs\4D Workshop 4 IDE\DEP
REM location of the script file
SET SCRIPT="D:\Diablo16\LoadProgram"
REM location of the program file
SET PROGRAM="D:\Diablo16\HelloWorld"
REM location of the COM port used
SET COMPORT=6

:Start

REM running SCRIPTC
"scriptc.exe" %SCRIPT% /ccom%COMPORT% /dprogramlocation %PROGRAM%

ECHO ****************************************
ECHO Would you like to program another module?
REM CHOICE /C YN /M "Press Y for Yes, or N for No"
SET /P RESULT=Y or N?
ECHO %RESULT%
IF "%RESULT%"=="n" GOTO Label2
IF "%RESULT%"=="N" GOTO Label2
IF "%RESULT%"=="y" GOTO Label1
IF "%RESULT%"=="Y" GOTO Label1
IF "%RESULT%"!="n" GOTO Label2
IF "%RESULT%"!="N" GOTO Label2
IF "%RESULT%"!="y" GOTO Label2
IF "%RESULT%"!="Y" GOTO Label2
```

```
:Label1
ECHO ****************
ECHO Repeating Process
GOTO Start

:Label2
ECHO **************
ECHO Ending Process
GOTO End

:End
```

10.1.2. 4D Script File

The batch file script assumes that the 4D script file “LoadProgram.4DScript” is located in "D:\Diablo16", as per the line

```
SET SCRIPT="D:\Diablo16\LoadProgram"
```

Modify the above line in the batch file script if necessary.

10.1.3. Program Files

The batch file scro[t assumes that the program files “HelloWorld.4XE" and “HelloWorld.cfg” are in the folder “D:\Diablo16”, as per the line

```
SET PROGRAM="D:\Diablo16\HelloWorld"
```

Modify the above line in the batch file script if necessary.

10.1.4. Comm Port

The batch file script assumes that the display module is connected to comm port COM6, as per the line

```
SET COMPORT=6
```

Connect the display module to the PC. Modify the batch file script and put the correct comm port if needed.

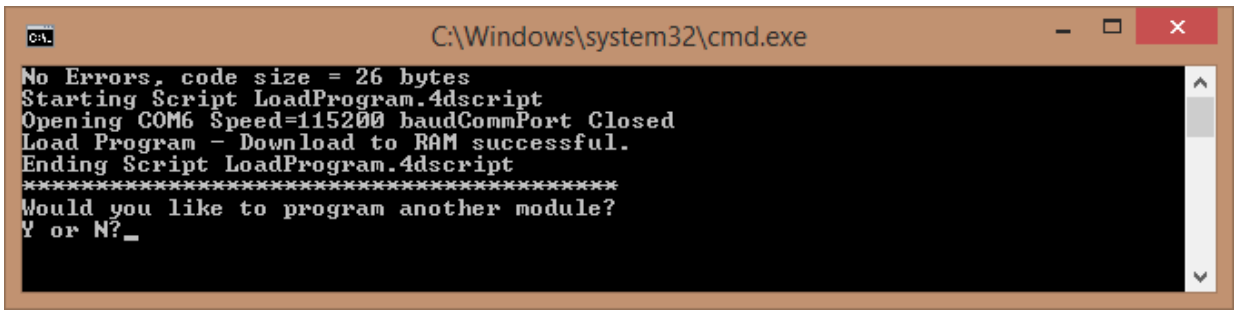
10.2. Run the Script

10.2.1. Execution

Click on the batch file to run it

10.2.2. Output

The console window should now open and print the results.



11. Appendix G: Return Codes

Return Code	Description
1	Help /? option used
2	Invalid parameter
4	Script file does not exist
5	Com Port not specified
6	Com Port Speed not specified
10	Fatal error, incorrect mode, type, or mutually exclusive options
100	Script aborted, check log for details

12. Revision History

Revision	Document Date	Description
1.0	9/10/2018	First draft
2.0	05/09/2019	Cosmetics
2.1	04/09/2020	Addition of Pixxi comments, and addition of missing return codes

13. Legal Notice

Proprietary Information

The information contained in this document is the property of 4D Labs Semiconductors and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Labs Semiconductors endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Labs Semiconductors products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Labs Semiconductors. 4D Labs Semiconductors reserves the right to modify, update or makes changes to Specifications or written material without prior notice at any time.

All trademarks belong to their respective owners and are recognised and acknowledged.

Disclaimer of Warranties & Limitation of Liability

4D Labs Semiconductors makes no warranty, either expressed or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

Images and graphics used throughout this document are for illustrative purposes only. All images and graphics used are possible to be displayed on the 4D Labs Semiconductors range of products, however the quality may vary.

In no event shall 4D Labs Semiconductors be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Labs Semiconductors, or the use or inability to use the same, even if 4D Labs Semiconductors has been advised of the possibility of such damages.

4D Labs Semiconductors products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Labs Semiconductors and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Labs Semiconductors' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Labs Semiconductors from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Labs Semiconductors intellectual property rights.

14. Contact Information

For Technical Support: www.4dlabs.com.au/support
For Sales Support: sales@4dlabs.com.au
Website: www.4dlabs.com.au