

Using the Scripting Utility in the Code Composer Studio Integrated Development Environment

Nipuna Gunasekera

Software Development Systems/Customer Support

ABSTRACT

Scripting is a new utility available for the Code Composer Studio[™] integrated development environment (IDE) v2.10. It provides a library of commands that integrate into Perl or Visual Basic for Applications (VBA), to provide a batch-mode scripting capability that can be utilized in automated testing and/or validation. This application report describes how to use scripting, in collaboration with Perl and VBA scripts, that will automatically invoke Code Composer Studio and perform some simple tasks.

Contents

1	Introduction	1
2	System Requirements	2
3	Scripting Using VBA Scripts	2
4	Scripting Using Perl Scripts	5
5	Items of Special Note	7
6	Reference	7

List of Examples

Example 1.	VBA Script Used With Scripting	4
Example 2.	Perl Script Used With Scripting	6

1 Introduction

Scripting is an interface that provides access from Perl or Visual Basic for Applications (VBA) to basic functions within Code Composer Studio. This application report describes how to use scripting, in collaboration with Perl and VBA scripts, that will automatically invoke Code Composer Studio and perform some simple tasks. For more detailed technical information, refer to the documentation available at (*Your_Install*)\ti\plugins\generic\CC_SCRIPTING.HLP.

Using scripting, a simple Perl or VBA script can load a program, set breakpoints, run, read/write memory and registers, execute general extension language (GEL) commands, and perform various other simple debugging functions.

Code Composer Studio, TMS320C5000, TMS320C6000 and OMAP are trademarks of Texas Instruments.

Trademarks are the property of their respective owners.



2 System Requirements

Scripting requires the following setup:

- Code Composer Studio v 2.10 for the TMS320C5000[™] or TMS320C6000[™] digital signal processors (DSPs), or OMAP[™] DSP-based processor.
- Visual Basic for Applications capable program (e.g., MS Excel)
- Perl 5.004 is provided with Scripting.

3 Scripting Using VBA Scripts

Scripting provides VBA script examples located at (Your_Install)\ti\bin\utilities\ccs_scripting \examples\vba, which can be opened using Microsoft Excel. The following dialog box will appear, as it does on any Excel spreadsheet that has embedded VBA.

1. To open any one of the Excel files, double-click on the file.

Microsoft Excel		? ×
C:\ti\bin\utilities\ccs_scriptir macros.	ng\examples\vba\AppNote	Project.xls contains
Macros may contain viruses macros are legitimate, you r	. It is always safe to disab might lose some functional	le macros, but if the ity.

- 2. Click the Enable Macros button.
- 3. Click on Tools -> Macro -> Visual Basic Editor to see the actual Visual Basic code. The code can be executed by using the blue triangular button on the Excel toolbar.

To create a new VBA script:

 Open up Microsoft Excel and click on Tools -> Macro -> Macro, provide a Macro name, and click Create. If you wish, you can begin by copying and pasting the following script provided in the main module.

Mi	icro	soft \	Visua	al Ba	sic -	Finall	est.sls	[Mod	lule1	(Code))						
~~ !	Eile	<u>E</u> dit	⊻iev	i Ins	ert I	Format	Depnd	<u>R</u> un	Tools	<u>A</u> dd-In	is <u>W</u> i	ndov	v <u>H</u> €	elp		
	13	• [() டி	Ĉ	#	NG CH		I	• 🖄	Bi	P	*	R	2	Ln 2, 0
(Ge	ener	al)														•
	R	efere	nce	s - VI	3APr	oject										X
		<u>A</u> vaila	ble R	efere	nces:								Ĩ	0	<	1
		Vis	sual B	asic F	or Ap	plicatio	ins				1			Can	cel	
			croso E Au	ft Exc tomat	el 9.0 ion) Objec	t Library				1					-
			croso IS So	ft Off riptin	ice 9. g. Cor	0 Obje n 1.0 1	ct Library 'ype Librar	y						Brows	se	
			ide C tive :	ompo Setup	ser 1. Conti	2 type rol Libr	library ary					t				
			tiveN tiveX	lovie Conf	contro erenc	ol type e Cont	library rol				Pric	ority		Нe	lp .	1
			tiveX	DLL t Type	o per : Libra	form M irv	igration of	MS R	eposit	ory V1		ŧ				
		Bio	osCor i 2.0	nfigSv Tvpe	r 1.0 Librar	Type L v	ibrary									
		∏ Bt	i2000	i.o1	VDe L	ibrarv				•	1					
		-Cod	e Cor	npose	r 1.2	type lil	orary				÷					÷
			Lo	, catior	n C	 ::\ti\cc'	, bin\cc_ap	p.exe								
			La	nguaq	je: :	Standa	rd									
	- 8	enn	<u>n a</u>	100		Te .	SPET	ne c	nrn		TTS	MB -	The	- 50	rint	-
	Ca	11 1	MyC	Scr	ipt	ing.	Script	Trac	eBeg	jin(Lo	ogFi	le)	one		L L P.	, ing c
	Ca	11 1	MyC	CScr	ipt	ing.	Script	Trac	eVei	bose	(CCS	_sc	RIP	TIN	G_CC	MLib.
	'C	lear	r ex	cist	ing	con	figura	tion	s							
	'I	npoi	nyC(rt (.ser 5551	ipt O s	ing. imul	ccscon ator c	tigC onfi	gura	tion						
	Ca	11 1	MyCO	Ser	ipt	ing.	CCSCon	figI	mpor	:t("c:	\ti	\dr	ive	rs\	impo	ort\c5

- 2. Make sure to create new references by clicking on Tools -> Reference, and check the CCS Scripting Com 1.0 Type and Code Composer 1.2 type libraries.
- 3. If you're using the following script as an example, make sure to save the newly created .xls file in (*Your_Install*)\ti\bin\utilities\ccs_scripting\examples\vba, as it uses one of the example projects in that directory.



Example 1 assumes that you have C5000[™] Code Composer Studio v2.1 or higher installed along with Code Composer Studio Scripting. This script assumes that the TMS320C5510 simulator has been configured via CC_Setup. The script will load and run one of the example programs packaged with Code Composer Studio Scripting, and also will demonstrate the method for setting breakpoints and reading register values.

Example 1. VBA Script Used With Scripting

```
Sub CCStudio Scripting()
'Declarations
'Create a new CCStudio Scripting object
Dim MyCCScripting As New CCS SCRIPTING COMLib.CCS Scripting
Dim MyPath As String
Dim MyProgram As String
Dim Visible As Integer
Visible = 1
MyPath = ThisWorkbook.Path
MyProgram = MyPath + "\hello\hello.out"
MyProject = MyPath + "\hello\hello.pjt"
'The following test:
'Opens the currently configured CCStudio
'Opens and Builds a project
'Loads a program
'Sets a breakpoint at "main"
'Runs to "main"
'Reads the value of the PC
'Closes the CCStudio application
Call MyCCScripting.CCSOpenNamed("*", "*", Visible)
Call MyCCScripting.ProjectOpen(MyProject)
Call MyCCScripting.ProjectBuild
Call MyCCScripting.ProgramLoad(MyProgram)
MainVal = MyCCScripting.SymbolGetAddress("main")
Call MyCCScripting.BreakpointSetAddress(MainVal)
Call MyCCScripting.TargetRun
MyPCVal = MyCCScripting.RegisterRead("PC")
Call MyCCScripting.CCSClose
If MyPCVal = MainVal Then
   Cells(4, 4) = "Test Passed"
End If
```

End Sub

NOTE: When a VBA program makes a Scripting call (such as TargetRun) that may take several minutes, the VBA program may show a message such as "Excel waiting for OLE Application" or "Server Busy." To disable these warnings, place the statement "Application.DisplayAlerts = False" into your VBA script before the Scripting calls.

C5000 is a trademark of Texas Instruments.

4 Scripting Using Perl Scripts

Scripting will automatically install Perl 5.004 in

(Your_Install)\ti\bin\utilities\ccs_scripting\perl

Perl examples, packaged with Scripting, are available in
(Your_Install)\ti\bin\utilities\ccs_scripting\examples\perl

To run any one of the examples:

Command Prompt
C:\ti\bin\utilities\ccs_scripting>CCS_Scripting_Perl.bat
C:\ti\bin\utilities\ccs_scripting>set path=c:\ti\bin\utilities\ccs_scripting\per l\bin;c:\ti\bin\utilities\ccs_scripting;"c:\ti\bin\utilities\ccs_scripting\perl\ bin;c:\ti\bin\utilities\ccs_scripting;"C:\WINNT\system32;C:\WINNT;C:\Program Fil es\Maestro.nt\;C:\ti\C6000\DSK\BIN;;C:\ti\bin;C:\ti\c6000\cgtools\bin;;C:\ti\c60 00\evn6x\bin;C:\tic2xx\c2000\cgtools\bin;""
C:\ti\bin\utilities\ccs_scripting>set PERL5LIB=;c:\ti\ccs_scripting\perl\lib;c:\ ti\bin\utilities\ccs_scripting;c:\ti\ccs_scripting\perl\lib;c:\ti\bin\utilities\ ccs_scripting C:\ti\bin\utilities\ccs_scripting>

- Run the CCS_Scripting_Perl.bat file which is provided in (Your_Install)\ti\bin\utilities\ ccs_scripting\. This will set the PATH and PERL5LIB environment variables properly for executing Perl scripts, as shown by the above image. This .bat file may be run as-is, or the paths referenced may be added to your system environment. This .bat file must be executed even if you have Perl5.004 or Perl5.005 on your machine, as it sets up your environment to include the needed Scripting files.
- 2. Run any of the Perl scripts using the following command. In this case, Debug.pl is the actual script being executed.

(Your_Install)\ti\bin\utilities\ccs_scripting\examples\perl>perl ccs_Debug.pl

If you wish, you may start by copying and pasting the following script provided as a *.pl file. If you are using the following script as an example, make sure to save the newly created *.pl file in *(Your_Install)*\ti\bin\utilities\ccs_scripting\examples\perl, as it uses one of the example projects in that directory.

Example 2 assumes that you have C5000 Code Composer Studio v2.1 or higher installed along with Scripting, and also assumes that a C5510 simulator is the sole existing configuration. This will load and run one of the example programs packaged with Code Composer Studio.



Example 2. Perl Script Used With Scripting

```
#Example of a Perl script used with CCStudio Scripting:
use CCS_SCRIPTING_PERL;
#Declarations
#Create a new CCStudio Scripting object
my $MyCCScripting = new CCS_SCRIPTING_PERL::CCS_Scripting();
my $MyProgram;
my $MyPath;
my $MyPCVal;
my $MainVal;
my $LogFile;
$MyPath = "C:\\ti\\bin\\utilities\\ccs_scripting\\examples\\perl";
$MyProgram = ".\\hello\\hello.out";
$MyProject = ".\\hello\\hello.pjt";
$LogFile = ".\\Test.log";
#Begin a log file. Set the output to trace the Scripting calls made
$MyCCScripting -> ScriptTraceBegin($LogFile);
#The following test:
#Opens the currently configured CCStudio
#Opens and Builds a project
#Loads a program
#Sets a breakpoint at "main"
#Runs to "main"
#Reads the value of the PC
#Closes the CCStudio application
$MyCCScripting -> CCSOpenNamed("*","*",1);
$MyCCScripting -> ProjectOpen($MyProject);
$MyCCScripting -> ProjectBuild();
$MyCCScripting -> ProgramLoad($MyProgram);
$MainVal = $MyCCScripting -> SymbolGetAddress("main");
$MyCCScripting -> BreakpointSetAddress($MainVal);
$MyCCScripting -> TargetRun();
$MyPCVal = $MyCCScripting -> RegisterRead("PC");
$MyCCScripting -> CCSClose();
if ($MyPCVal == $MainVal)
{
    $MyCCScripting -> ScriptTraceWrite("Test Passed \n\n");
}
```

NOTE: Any **GEL** command may be executed in Scripting via the TargetEvalExpression applications programming interface (API). For example, mapping memory from Gel in Scripting would look like the following:

5 Items of Special Note

Code Composer Studio may be opened one of two ways:

 CCSOpenNamed(sBoardName, sCPUName, bVisible) will open the specific board and CPU named by the strings sBoardName and sCPUName, as they have been named in the configuration in CC_Setup.

A wildcard character "*" may be given for sBoardName and sCPUName. For example, CCSOpenNamed("*", "*", bVisible) will open the first or only item in the configuration.

 CCSOpen(nMajorISA, nMinorISA, nCPUIndex, nPlatform, bVisible): This opens a specific board or a simulator from the configuration list.

CCSOpen allows the user to specify which CPU in a particular debug chain to open through the third parameter, nCPUIndex.

nCPUIndex is a zero-based index. For example, nCPUIndex = 1 specifies the second CPU of the designated type found.

Examples in VBA:

Open a c55x simulator if it is the only one in the configuration list:

```
Call MyScript.CCSOpenNamed("*", "*", true)
```

Open 2 c55x emulators from a configuration list simultaneously using CCSOpen:

Call MyScript.CCSOpen(CCS_SCRIPTING_COMLib.ISA_C55, 0, 0, CCS_SCRIPTING_COMLib.PLATFORM_EMULATOR, true)

Call MyScript.CCSOpen(CCS_SCRIPTING_COMLib.ISA_C55, 0, 1, CCS_SCRIPTING_COMLib.PLATFORM_EMULATOR, true)

Open 2 c55x emulators from a configuration list simultaneously using CCSOpenNamed:

Call MyScript.CCSOpenNamed("C55x Texas Instruments", "CPU_1", true)

- Call MyScript.CCSOpenNamed("C55x Texas Instruments", "CPU_2", true)
- CCSClose takes in a parameter, nCloseAll, which default to true (1). When nCloseAll = true, CCSClose will terminate all Code Composer Studio processes that are running. When nCloseAll = false, only the current Code Composer Studio processes will be closed. It is highly recommended that CCSClose(true) be specified when all scripting functions have been completed, to avoid leaving orphaned processes running on the system.
- If a user script is interrupted by an error condition (e.g., "ProgramLoad: file not found"), when your script shuts down, make sure that there is no cc_app.exe processes still running in the background. Having an orphaned cc_app running when the Scripting functions are called can lead to unpredictable results.

6 Reference

1. Scripting documentation

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments Post Office Box 655303 Dallas, Texas 75265

Copyright © 2002, Texas Instruments Incorporated