

Script commands in CodeScape Version 2.2.0 Build 111

Write scripts to automate tasks

CodeScape's script commands let you run Microsoft® JScript™ and VBScript macro scripts to automate routine tasks.

CodeScape's script commands are demonstrated in the example JScript and VBScript files included in this document. Use the functions available in either script language to add your own commands.

For details about using JScript and VBScript connect to the scripting area on the Microsoft Developer Network at <http://msdn.microsoft.com/scripting>

CodeScape's script commands

Note: Some scripting engines reserve the use of Write, use CodeScape's WriteMessage function when you need the Write function.

Description	Syntax
Load the specified program file. This command uses the file path as a parameter and returns 1 if the file is loaded, else 0.	LoadProgramFile(path and filename)
Reset the target processor with a hard reset.	HardReset()
Reset the target processor with a soft reset.	SoftReset()
Run the target processor.	Run()
Write a message string to the script window.	WriteMessage(string Message)
Set the specified register to the given value.	WriteRegister(Register value, Numeric value)
Get the value held in the specified register.	RegisterValue ReadRegister(RegisterName)
Load a binary file from the specified location.	LoadBinaryFile(Path and filename, Numeric binary location)
Set a code breakpoint at the specified address.	SetBreakpoint(Numeric address)
Clear all breakpoints.	ClearAllBreakpoints()
Remove the breakpoint from the specified address.	RemoveBreakpoint(Numeric address)
Read a byte from the specified area of memory.	ReadByte(Numeric address)
Read a word from the specified area of memory.	ReadWord(Numeric address)
Read a long from the specified area of memory.	ReadLong(Numeric address)
Write a byte from the specified area of memory.	WriteByte(Numeric address, Numeric value)

Description	Syntax																								
Write a word from the specified area of memory.	WriteWord(Numeric address, Numeric value)																								
Write a long from the specified area of memory.	WriteLong(Numeric address, Numeric value)																								
Return a specific parameter.	GetParam(short param)																								
Returns the number of parameters passed to the script.	GetParamCount()																								
Return 1 if running, 0 if not running.	IsRunning()																								
Specify the events saved in the Trace history: <table><tr><td>Events</td><td>Setting</td></tr><tr><td>Log exceptions, interrupts, and rte</td><td>8</td></tr><tr><td>Log subroutines, bsr, bsrf, jsr, rts</td><td>4</td></tr><tr><td>Log branches, bf, bt, bf/s, bt/s, bra, braf,jmp</td><td>2</td></tr></table>	Events	Setting	Log exceptions, interrupts, and rte	8	Log subroutines, bsr, bsrf, jsr, rts	4	Log branches, bf, bt, bf/s, bt/s, bra, braf,jmp	2	ConfigureTraceHistory(numeric Setting, boolean Enable)																
Events	Setting																								
Log exceptions, interrupts, and rte	8																								
Log subroutines, bsr, bsrf, jsr, rts	4																								
Log branches, bf, bt, bf/s, bt/s, bra, braf,jmp	2																								
Display the current history in the script's window in this format: <table><tr><td>Source</td><td>Destination</td><td></td></tr><tr><td>0x0c010356</td><td>0x0c0103aa</td><td>rts</td></tr><tr><td>0x0c0101e6</td><td>0x0c010350</td><td>rts</td></tr><tr><td>0x0c0100e6</td><td>0x0c010128</td><td>bra \$0c010128</td></tr><tr><td>0x0c01034c</td><td>0x0c010028</td><td>bsr BigTest</td></tr><tr><td>0x0c0103a6</td><td>0x0c010334</td><td>bsr struct_test</td></tr><tr><td>0x0c010280</td><td>0x0c0103a0</td><td>rts</td></tr><tr><td>0x0c01039c</td><td>0x0c010214</td><td>bsr BitFieldTest</td></tr></table>	Source	Destination		0x0c010356	0x0c0103aa	rts	0x0c0101e6	0x0c010350	rts	0x0c0100e6	0x0c010128	bra \$0c010128	0x0c01034c	0x0c010028	bsr BigTest	0x0c0103a6	0x0c010334	bsr struct_test	0x0c010280	0x0c0103a0	rts	0x0c01039c	0x0c010214	bsr BitFieldTest	DisplayTraceHistory()
Source	Destination																								
0x0c010356	0x0c0103aa	rts																							
0x0c0101e6	0x0c010350	rts																							
0x0c0100e6	0x0c010128	bra \$0c010128																							
0x0c01034c	0x0c010028	bsr BigTest																							
0x0c0103a6	0x0c010334	bsr struct_test																							
0x0c010280	0x0c0103a0	rts																							
0x0c01039c	0x0c010214	bsr BitFieldTest																							
Clear the script output window.	ClearDisplay()																								
Create a breakpoint of the given type at the address. Returns a breakpoint identifier on success, otherwise 0. The breakpoint identifier is used in subsequent operations on the breakpoint. <table><tr><td>Breakpoint</td><td>Type</td></tr><tr><td>Code</td><td>0</td></tr><tr><td>Watch</td><td>1</td></tr><tr><td>Simulator or Start</td><td>2</td></tr><tr><td>Profiler start</td><td>3</td></tr><tr><td>Profiler stop</td><td>4</td></tr></table>	Breakpoint	Type	Code	0	Watch	1	Simulator or Start	2	Profiler start	3	Profiler stop	4	CreateBreakpoint(Type, Address)												
Breakpoint	Type																								
Code	0																								
Watch	1																								
Simulator or Start	2																								
Profiler start	3																								
Profiler stop	4																								
Enable or disable the breakpoint: identifier: the breakpoint identifier. enable: 1 to enable; 0 to disable.	EnableBreakpoint(identifier, boolean enable)																								
Enable or disable the specified breakpoint action: enable: 1 to enable , 0 otherwise. identifier: the breakpoint identifier. <table><tr><td>Action</td><td>Value</td></tr><tr><td>Halt breakpoint when hit.</td><td>0</td></tr><tr><td>Remove breakpoint after being hit.</td><td>1</td></tr><tr><td>Display a message box prompt when hit</td><td>2</td></tr><tr><td>Beep when hit</td><td>3</td></tr></table>	Action	Value	Halt breakpoint when hit.	0	Remove breakpoint after being hit.	1	Display a message box prompt when hit	2	Beep when hit	3	SetBreakpointActions(identifier, numeric action, boolean enable)														
Action	Value																								
Halt breakpoint when hit.	0																								
Remove breakpoint after being hit.	1																								
Display a message box prompt when hit	2																								
Beep when hit	3																								
Set a log expression for the breakpoint specified by the breakpoint identifier: breakpoint identifier: the breakpoint identifier. expression: the log expression. logType: false to always log or true to log when conditions match.	SetBreakpointLog(breakpoint identifier, string expression, boolean logType)																								

Description	Syntax
Attach a script to a breakpoint: identifier: the breakpoint identifier. script path: the file path for the script script type: 0 for JScript and1 for VBScript script arguments: string holding the script's arguments prompt: 1 to request arguments when the breakpoint triggers, 0 otherwise.	SetBreakpointScript(identifer, string script path, numeric script type, string script arguments, boolean prompt)
Set the conditional expression for the breakpoint: identifier: the breakpoint identifier. expression: a string representing the condition. expression type: 0 for C; non-zero for assembly trigger count: the number of hits before breakpoint actions are performed. incOnTrue: false to always increment the trigger count; true to increment the trigger count only when conditions are true. breakWhen: false to break when the trigger reaches 0 or condition is true; true to break when trigger reaches zero and the condition is true.	SetBreakpointCondition(identifier, string expression, numeric expression type, numeric trigger count, boolean incOnTrue, boolean breakWhen)
Set the parameters for a watch breakpoint: identifier: the breakpoint identifier. incDataCondition: include a data condition. dataCondition: expression specifying the data condition. expressionType: the type of the specified expression. accessSize: the access size e.g. byte, word, or long etc. accessType: the type of access (read, write, or both). Size Value Type Value Any 0 Read 1 Byte 1 Write 2 Word 2 Read or Write 3 Long 4 Quad 8	BOOL SetWatchBreakpointParameters(Identifier, Boolean incDataCondition, string dataCondition, numeric expressionType, numeric accessSize, numeric accessType)
Select a location mask for the breakpoint: Mask Value No bits masked 1 Lower 10 bits 2 Lower 12 bits 3 Lower 16 bits 4 Lower 20 bits 5 All bits 6	SetBreakpointLocationMask(breakID, maskSelect)
Set the data mask for a watch breakpoint	SetBreakpointDataMask(identifier, mask)

You can express Numeric values and Numeric addresses as:

- A number, for example, 124 or 3.1415926
-OR-
- A string, for example, "124"
-OR-
- Hexadecimal in a string, for example, "0xabcdef"
-OR-
- A symbol, for example, "main" or "main + 0xabc"

Note: Registers are only passed as strings, for example, "pc", "fr0", "r0".

Note: Currently, scripts only support debugging a single target processor. When you run a script it automatically uses the selected target processor.

Note: A script that contains an infinite loop will cause CodeScape to lock-up.

Example VBScript

```
' This script does not do anything useful other than demonstrate the functions available
```

```
ClearDisplay
HardReset
SoftReset
DisplayParameters
LoadSomeBinary
LoadProgramFile( "d:\\projects\\maketest\\hello.elf" )
SetBreakpoint( "add_fn" )
ConfigureTraceHistory TH_LOGEXCEPT + TH_LOGSUB, true
Dim Running
Running = 1
Do
    Running = IsRunning
Loop Until Running = 0
DisplayTraceHistory
ReadSomeRegisters
WriteSomeRegisters
ReadSomeMemory
WriteSomeMemory
ReadSomeMemory
ClearAllBreakpoints
CreateCodeBP
ClearAllBreakpoints
CreateWatchBP
WriteMessage( "Script complete. Removing all breakpoints." )
ClearAllBreakpoints
```

```
'
'      Breakpoint types
'
BPTYPE_CODE           =      0
BPTYPE_WATCH          =      1
BPTYPE_SIMSTART       =      2
BPTYPE_PROFSTART      =      3
BPTYPE_PROFSTOP       =      4

'
'      Breakpoint Actions
'
BPACTION_HALT         =      0
BPACTION_ONESHOT      =      1
BPACTION_PROMPT       =      2
BPACTION_BEEP         =      3

'
'      Breakpoint Script Types
'
BPSCRIPT_JSCRIPT      =      0
BPSCRIPT_VBSCRIPT     =      1

'
'      Breakpoint expression types
'
BPEXPR_C              =      0
BPEXPR_ASSEMBLY       =      1

'
'      Breakpoint address masks
'

BPLOCMASK_NONE        =      1
BPLOCMASK_LOW10       =      2
BPLOCMASK_LOW12       =      3
BPLOCMASK_LOW16       =      4
BPLOCMASK_LOW20       =      5
BPLOCMASK_ALL         =      6
```

```

'
'      Breakpoint access sizes
'

BPACCESSSIZE_ANY      =      0
BPACCESSSIZE_BYTE     =      1
BPACCESSSIZE_WORD     =      2
BPACCESSSIZE_LONG     =      4
BPACCESSSIZE_QUAD     =      8

'
'      Breakpoint access types
'

BPACCESSTYPE_READ     =      1
BPACCESSTYPE_WRITE    =      2
BPACCESSTYPE_RW       =      3

'
'      Trace history configuration options
'

TH_LOGEXCEPT =      8
TH_LOGSUB      =      4
TH_LOGBRANCH  =      2

'
'      Create a breakpoint on the 1K aligned block of memory that
'      the symbol main resides in.
'

Sub CreateCodeBP()

    Dim breakID

    breakID      =      CreateBreakpoint( BPTYPE_CODE, "main" )

    SetBreakpointAction breakID, BPACTION_HALT, true

    SetBreakpointAction breakID, BPACTION_ONESHOT, false

    SetBreakpointAction breakID, BPACTION_PROMPT, false
    SetBreakpointAction breakID, BPACTION_BEEP, true
    SetBreakpointScript breakID, "e:\\projects\\codescape\\debugs\\testscript.js",
BPSCRIPT_JSCRIPT, "arg1 arg2 arg3", false
    SetBreakpointLog breakID, "Hello John", BPEXPR_C
    SetBreakpointLocationMask breakID, BPLOCMASK_LOW10
    setBreakpointCondition breakID, "index == 375", BPEXPR_C, 37, true, true
End Sub

Sub CreateWatchBP()
    breakID      = CreateBreakpoint( BPTYPE_WATCH, "main" )
    SetWatchBreakpointParameters breakID, true, "14", BPEXPR_C, BPACCESSSIZE_BYTE,
BPACCESSTYPE_WRITE
End Sub

Sub WriteSomeRegisters()
    WriteRegister "fr0", 3.14159
    WriteRegister "r0", "0xabcdef"
    WriteRegister "pc", "main + 0x30"
End Sub

Sub ReadSomeRegisters()
    WriteMessage( "Value of pc = " & ReadRegister( "pc" ) )
    WriteMessage( "Value of r0 = " & ReadRegister( "r0" ) )
End Sub

```

```
Sub LoadSomeBinary()  
    LoadBinaryFile "d:\\projects\\codescape\\satmon.bin", "201392128"  
    LoadBinaryFile "d:\\projects\\codescape\\satmon.bin", 201392128  
    LoadBinaryFile "d:\\projects\\codescape\\satmon.bin", "0xc010000"  
    LoadBinaryFile "d:\\projects\\codescape\\satmon.bin", "main"  
End Sub  
  
Sub DisplayParameters()  
    NumParams = GetParamCount  
    WriteMessage( "Number of parameters = " & NumParams )  
    For i = 1 To NumParams  
        WriteMessage( "Parameter " & i & " = " & GetParam( i - 1 ) )  
    Next  
End Sub  
  
Sub ReadSomeMemory()  
    WriteMessage( "Byte at main = " & ReadByte( "main" ) )  
    WriteMessage( "Word at main + 4 = " & ReadWord( "main + 4" ) )  
    WriteMessage( "Long at main + 8 = " & ReadLong( "main + 8" ) )  
End Sub  
  
Sub WriteSomeMemory()  
    WriteByte "main", 255  
    WriteWord "main + 4", "0xabcd"  
    WriteLong "main + 8", "0xfedcba"  
End Sub
```

Example JScript

```

//      Note: this script does not do anything useful. It just demonstrates the current
//      script commands and how they can be called.
//
//      Breakpoint types
//
BPTYPE_CODE           =      0;
BPTYPE_WATCH          =      1;
BPTYPE_SIMSTART       =      2;
BPTYPE_PROFSTART      =      3;
BPTYPE_PROFSTOP       =      4;

//
//      Breakpoint Actions
//
BPACTION_HALT         =      0;
BPACTION_ONESHOT      =      1;
BPACTION_PROMPT       =      2;
BPACTION_BEEP         =      3;

//
//      Breakpoint Script Types
//
BPSCRIPT_JSCRIPT      =      0;
BPSCRIPT_VBSCRIPT     =      1;

//
//      Breakpoint expression types
//
BPEXPR_C              =      0;
BPEXPR_ASSEMBLY       =      1;

//
//      Breakpoint address masks
//
BPLOCMASK_NONE        =      1;
BPLOCMASK_LOW10       =      2;
BPLOCMASK_LOW12       =      3;
BPLOCMASK_LOW16       =      4;
BPLOCMASK_LOW20       =      5;
BPLOCMASK_ALL         =      6;

//
//      Breakpoint access sizes
//
BPACCESSSIZE_ANY      =      0;
BPACCESSSIZE_BYTE     =      1;
BPACCESSSIZE_WORD     =      2;
BPACCESSSIZE_LONG     =      4;
BPACCESSSIZE_QUAD     =      8;

//
//      Breakpoint access types
//
BPACCESSTYPE_READ     =      1;
BPACCESSTYPE_WRITE    =      2;
BPACCESSTYPE_RW       =      3;

//
//      Trace history configuration options
//
TH_LOGEXCEPT =      8;
TH_LOGSUB      =      4;
TH_LOGBRANCH  =      2;

```



```
//
//      Create breakpoint on the 1k aligned block of memory that the symbol main resides in
//
function CreateCodeBP()
{
    breakID      =      CreateBreakpoint( BPTYPE_CODE, "main" );
    SetBreakpointAction( breakID, BPACTIION_HALT, true );
    SetBreakpointAction( breakID, BPACTIION_ONESHOT, false );
    SetBreakpointAction( breakID, BPACTIION_PROMPT, false );
    SetBreakpointAction( breakID, BPACTIION_BEEP, true );
    SetBreakpointScript( breakID, "e:\\projects\\codescape\\debugs\\testscript.js",
BPSCRIPT_JSCRIPT, "arg1 arg2 arg3", false );
    SetBreakpointLog( breakID, "Hello John", BPEXPR_C );
    SetBreakpointLocationMask( breakID, BPLOCMASK_LOW10 );
    setBreakpointCondition( breakID, "index == 375", BPEXPR_C, 37, true, true );
}

function CreateWatchBP()
{
    breakID      =      CreateBreakpoint( BPTYPE_WATCH, "main" );
    SetWatchBreakpointParameters( breakID, true, "14", BPEXPR_C, BPACCESSSIZE_BYTE,
BPACCESSTYPE_WRITE );
}

function WriteSomeRegisters()
{
    WriteRegister( "fr0", 3.14159 );
    WriteRegister( "r0", "0xabcdef" );
    WriteRegister( "pc", "main + 0x30" );
}

function ReadSomeRegisters()
{
    WriteMessage( "Value of pc = " + ReadRegister( "pc" ) );
    WriteMessage( "Value of r0 = " + ReadRegister( "r0" ) );
}

function LoadSomeBinary()
{
    LoadBinaryFile( "d:\\projects\\codescape\\satmon.bin", "201392128" );
    LoadBinaryFile( "d:\\projects\\codescape\\satmon.bin", 201392128 );
    LoadBinaryFile( "d:\\projects\\codescape\\satmon.bin", "0xc010000" );
    LoadBinaryFile( "d:\\projects\\codescape\\satmon.bin", "main" );
}

function DisplayParameters()
{
    NumParams      =      GetParamCount()
    WriteMessage( "Number of parameters = " + NumParams );
    for( i = 0; i < NumParams; i++ )
    {
        WriteMessage( "Parameter " + i + " = " + GetParam( i ) )
    }
}

function ReadSomeMemory()
{
    WriteMessage( "Byte at main = " + ReadByte( "main" ) );
    WriteMessage( "Word at main + 4 = " + ReadWord( "main + 4" ) );
    WriteMessage( "Long at main + 8 = " + ReadLong( "main + 8" ) );
}

function WriteSomeMemory()
{
    WriteByte( "main", 255 );
    WriteWord( "main + 4", "0xabcd" );
    WriteLong( "main + 8", "0xfedcba" );
}
```

```
ClearDisplay();
HardReset();
SoftReset();
DisplayParameters();
LoadSomeBinary();
LoadProgramFile( "d:\\projects\\maketest\\hello.elf" );
SetBreakpoint( "add_fn" );
ConfigureTraceHistory( TH_LOGEXCEPT + TH_LOGSUB, true );
Run();
while( IsRunning() != 0 )
{
    ;
}
DisplayTraceHistory();
ReadSomeRegisters();
WriteSomeRegisters();
ReadSomeMemory();
WriteSomeMemory();
ReadSomeMemory();
ClearAllBreakpoints();
CreateCodeBP();
ClearAllBreakpoints();
CreateWatchBP();
WriteMessage( "Script complete. Removing all breakpoints." );
ClearAllBreakpoints();
```

Using scripts

When you run a script the Input / Output window appears automatically and displays the Script tab with all messages generated by the current script.

To open the Input / Output window without running a script:

- Click View, Toolbar, then select the Input / Output check-box and click OK.

Note: You can dock the Input / Output window at the top and bottom of the main window, or leave it free floating.

The shortcut menu on the Script tab

Click:	To:
Run Script	Select and run a script.
Clear	Clear the contents of the Script tab.
User Scripts	This option appears in gray until you add a script to the menu. When you add a script its name appears on the menu.
Allow Docking	Toggle docking for the window on or off.
Hide	Hide the window.

Add a script to the menu

When you add a script its name appears on the menu bar, and on the Script tab shortcut menu. You can add up to ten script files to run from either the menu bar, or the shortcut menu.

- 1 Click Tools, select Customize, then click Scripts...
The Customize dialog box appears.
- 2 Click Add.
- 3 In the Menu Text box, enter the script name to display on menu.
To remove an entry from the Menu Text box, select the script, then click Remove.
- 4 In the Menu Contents box, highlight the name of the script.
- 5 In the Script box, enter the path location and script file name.
- 6 Select either JScript, or VBScript to specify the script file type.
- 7 Do one of the following:
 - In the Arguments text box, enter any arguments to be passed to the script. Click OK.
 - OR–
 - Select the Prompt for arguments check-box.

Note: Select a command in the Menu Contents box, then Use Move Up and Move Down to set where it appears on the Tools menu.

Note: To assign a keyboard shortcut to the script click Tools, select Customize then click Keyboard...

Run a script

- Click Tools, then select Scripts and click a script in the list.
- OR-
- On the Input / Output window, right-click on the Scripts tab, then click a script in the list.

Note: Currently, scripts only support debugging a single target processor. When you run a script it automatically uses the selected target processor.

Note: A script that contains an infinite loop will cause CodeScape to lock-up.
