



ATC 2008

MSP430 Advanced Technical Conference

Advanced Debug Capabilities of CCE

Darian Sale

5/21/2008

1



Software Development Organization

Objectives

- Introduction to advanced features of CCE
 - Debug Server Scripting (DSS)
 - 3rd party Eclipse plug-ins
 - Other advanced Eclipse features
 - Perspectives and workspaces
 - Local history
 - Refactoring
 - Advanced breakpoint features



ATC 2008

MSP430 Advanced Technical Conference

2



DSS: What is it?



- Java API's to CCE debugger
- Lightweight package with no GUI
 - Faster, as there is no GUI to refresh/repaint/update
 - Errors handled via exceptions, not dialogs
 - However, specific GUI actions cannot be automated
- Scriptable through available 3P tools such as:
 - Javascript (via Rhino)
 - Perl (via "inline::java" module)
 - Python (via Jython)
 - TCL (via Jacl/Tclblend)

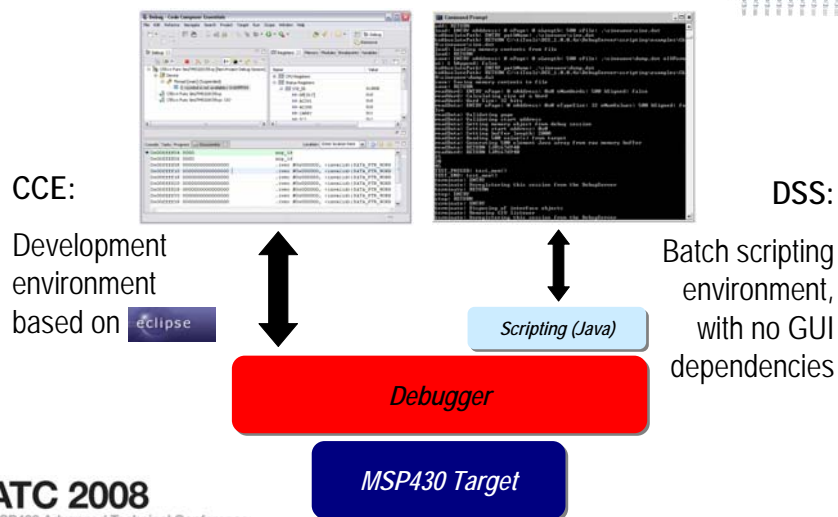
ATC 2008

MSP430 Advanced Technical Conference

3



DSS: What is it?



ATC 2008

MSP430 Advanced Technical Conference

4



DSS: What can you use it for?



- Automate testing
 - Use JUnit, tcltest etc
- Automate profile runs overnight
- Type commands instead of clicking a mouse
- Automate a sequence of complex steps
 - Quickly place the target application into a “broken” state for debugging

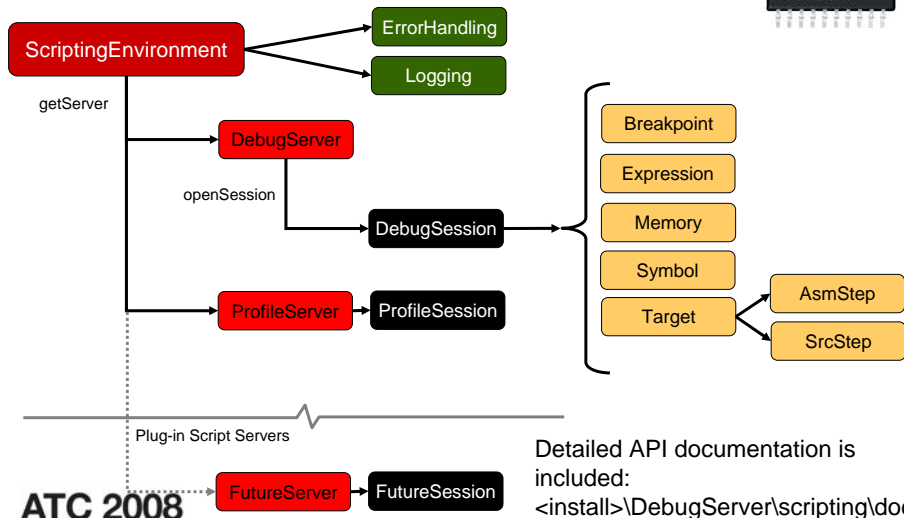
ATC 2008

MSP430 Advanced Technical Conference

5



DSS: API's



ATC 2008

MSP430 Advanced Technical Conference

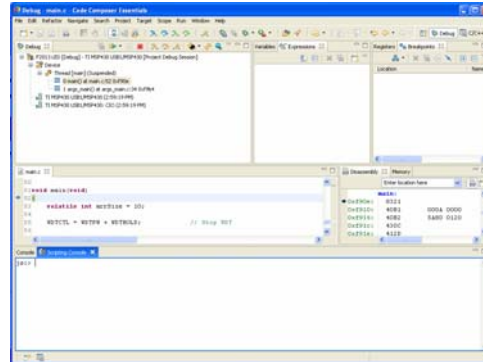
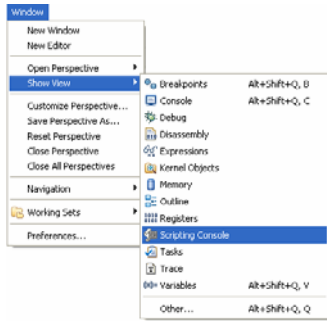
6



DSS: How to use it



- Used from command line or Scripting Console
 - The Scripting Console can be opened from the Window->Show View menu and executes javascript



ATC 2008
MSP430 Advanced Technical Conference

7



DSS: Scripting Console

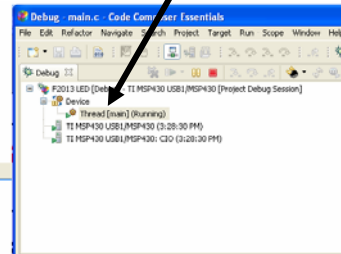


- Scripting Console creates a session variable *dss*
 - session opened from *dss* links with active debug session

Target starts to run

Run Issued

```
Scripting Console
js> var session = dss.openSession()
js> session.target.run()
```



ATC 2008
MSP430 Advanced Technical Conference

8

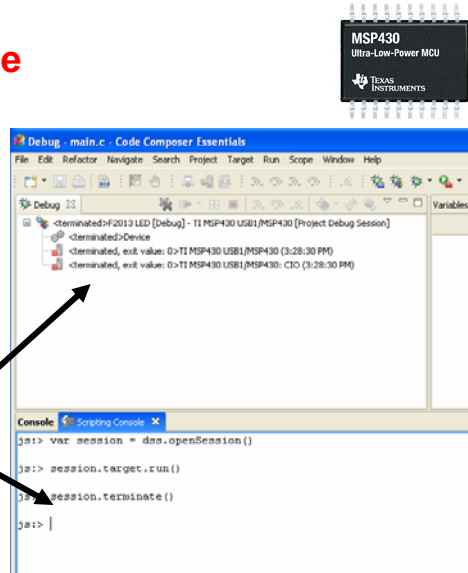


DSS: Scripting Console

- Terminating the session will terminate the CCE debug session

Terminate called

Session terminates



ATC 2008

MSP430 Advanced Technical Conference

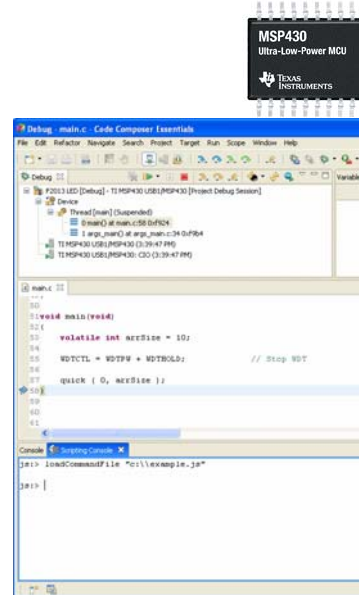
9



DSS: Scripting Console

- Use `loadCommandFile` to run a more complex script
 - Script will be run at every startup

```
1
2 // Get the active session
3 session = dss.openSession()
4
5 // Reset the target
6 session.target.reset()
7
8 // Set a breakpoint at the end of main
9 session.breakpoint.add( "main.c", 58 )
10
11 // Run the target
12 session.target.run()
```



ATC 2008

MSP430 Advanced Technical Conference

10

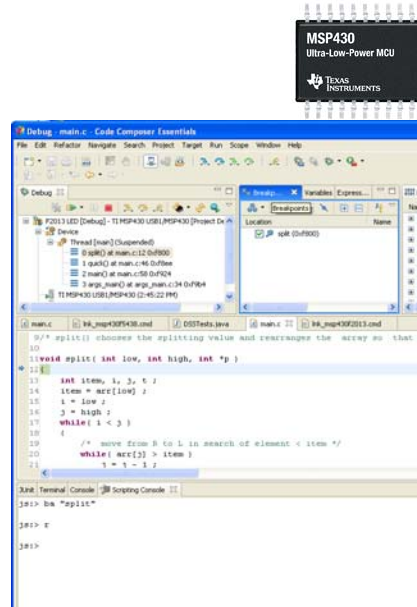


DSS: Scripting Console

- Place commonly used functions in a script
 - Allows easy keyboard control of the debugger

```
// Breakpoint Add at a sepecific address
function ba( address )
{
    session.breakpoint.add( address )
}

// Run the target
function r()
{
    session.target.run()
}
```



ATC 2008

MSP430 Advanced Technical Conference



DSS: Command line

- DSS can also be used from the command line
- In the CCE install folder, go to `\DebugServer\scripting\examples`
 - This folder contains rhino.bat, which will run any DSS javascript-based file
 - The folder also contains some example DSS scripts



ATC 2008

MSP430 Advanced Technical Conference



DSS: Sample script



```

// Import the DSS packages into our namespace to save on typing
importPackage( Packages.com.ti.debug.engine.scripting )
importPackage( Packages.com.ti.ccstudio.scripting.environment )
importPackage( Packages.java.lang )

// Create Scripting Environment
var env = new ScriptingEnvironment();

// Begin logging - log *everything* to a
// file and INFO messages to the console
env.traceBegin( "dsslog.xml" );
env.traceSetFileLevel( TraceLevel.ALL );
env.traceSetConsoleLevel( TraceLevel.INFO );

// Open a debug session
var server = env.getServer( "DebugServer.1" );
var session = server.openSession();

// Connect and load the .out file
try {
    session.target.connect();
    session.memory.loadProgram( "simple.out" );
} catch (ex){
    // Do Failure Routine
}

```

Scripting Environment encapsulates logging, error handling, etc.

Logging to console, file or combination of both

Try...Catch blocks are supported for robust error handling

ATC 2008

MSP430 Advanced Technical Conference

13



DSS: Sample script



```

// Set start & stop profiling BPs
session.breakpoint.add( "simple.c", 5 );
session.breakpoint.add( "simple.c", 20 );

// Run to first (start) breakpoint
session.target.run();

// Run to second breakpoint and count cycles
var profileServer = env.getServer( "ProfileServer.1" );
var profileSession = profileServer.openSession( session );
var cycleCount = profileSession.runBenchmark();

// Report how long the run took
env.traceWrite( "cycle.CPU (Incl. Total): " + cycleCount + "
cycles" );

// Close our Session and Server
session.terminate();
server.stop();

```

Simple to perform common debug operations

Synchronous run

Could perform multiple runs and compute Std. Dev. from acceptable norm value.

Terminate our debug session

ATC 2008

MSP430 Advanced Technical Conference

14



DSS: Logging



- XML format (easy to parse)
 - Can specify XSTL file to be referenced in XML log file
 - Configure how log information is displayed when opened in a web browser
 - XSTL Tutorial: http://www.w3schools.com/xsl/xsl_intro.asp

- More information
 - Sequence Numbers
 - Timing information (total, deltas, etc)
 - Status messages

- Log printf output

ATC 2008

MSP430 Advanced Technical Conference

15



DSS: Log file in web browser



Debug Server Log

Total Execution Time: 89517 ms

Sequence	Time (ms)	Delta (ms)	Level	Method	Message
0	0		FINER	traceSetConsoleLevel	RETURN
1	0	0	FINER	getServer	ENTRY sServerName: LegacySetupServer.1
2	0	0	FINER	getServer	Getting definition for: LegacySetupServer.1
3	10	10	FINER	getServer	Constructing server
4	10	0	FINER	getServer	Starting server
5	20	10	FINER	start	ENTRY
6	50	30	FINER	start	RETURN
7	50	0	FINER	getServer	RETURN com.ti.debug.engine.scripting.LegacySetupServer@253498
8	50	0	FINER	ccsConfigClear	ENTRY
9	50	0	FINER	ccsConfigClear	Creating system setup object
10	2865	2815	FINER	ccsConfigClear	Clearing configurations
11	4166	1301	FINER	ccsConfigClear	Saving system configuration
12	6260	2094	FINER	ccsConfigClear	RETURN
13	6260	0	FINER	ccsConfigImport	ENTRY sConfig C:\DSS\DebugServer\drivers\import\DM6446_itl_endian_sim.ccs

ATC 2008

MSP430 Advanced Technical Conference

16



DSS: Exception Handling



```
try {
    debugSession.memory.loadProgram(testProgFile);
} catch (ex){
    errCode = dssErrorHdl.getLastErrorID();
    dssScriptEnv.traceWrite("errCode: " + errCode + " - " +
        testProgFile + " does not exist!");
    quit();
}
```

- DSS APIs throw Java exceptions
- Can handle exception in the script to fail gracefully or continue on

ATC 2008

MSP430 Advanced Technical Conference

17



DSS: Example



- Demo
 - Use DSS to automate testing of an MSP430 application

ATC 2008

MSP430 Advanced Technical Conference

18



Eclipse background

- Eclipse
 - Originally IBM's Java Development Environment
 - Open platform for tool integration built by an open community of tool providers
 - Operates under a open source paradigm
 - Universal platform for tools integration
 - Multi-language, multi-platform and multi-vendor environment
 - Excellent plug-in framework
- Governed by Eclipse Public License (EPL)
 - Royalty free world-wide redistribution rights
 - Enables you to sell components & products
 - Source changes to EPL governed code need to be made available
- Eclipse Foundation
 - The Eclipse Foundation is a non-profit corporation formed to advance the creation, evolution, promotion, and support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities, and services.
- Other Eclipse members
 - IBM/Rational, QNX, WindRiver, Borland, RedHat, MontaVista, Texas Instruments



ATC 2008

MSP430 Advanced Technical Conference

19



Benefits of Eclipse

- Excellent Application Framework
 - Modular and extensible
- Rapid Rate of Improvement
 - Many companies are contributing
- Development tool integration
 - Users need more than TI tools to do their development
 - Design tools, Code development, OS awareness
 - www.eclipseplugincentral.com
 - www.eclipse.org/community/plugins.php
 - Eclipse is a standard platform to integrate with
 - Outstanding SDK and plug-in framework
- Host platform independence
 - Windows, Linux...
- TI can focus on unique value-added capabilities



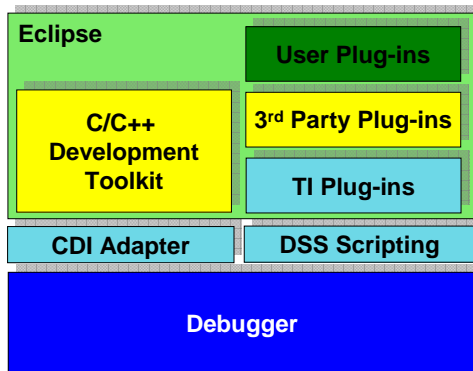
ATC 2008

MSP430 Advanced Technical Conference

20



CCE architecture



ATC 2008

MSP430 Advanced Technical Conference

21



Eclipse Plug-ins



- Well documented API enables 3rd parties
 - Many Eclipse add-ons available
 - Editors
 - Source control
 - Testing
 - Documentation
 - UML
- Caveat
 - Not all plugins will work in CCE, or work together

ATC 2008

MSP430 Advanced Technical Conference

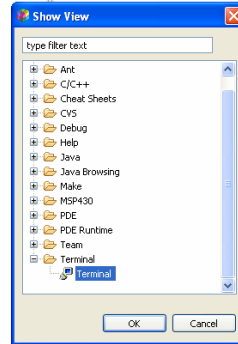
22



Example - Terminal plug-in



- CCE ships with the Terminal View plug-in
 - Available from the *Other...* menu in *Show View*



ATC 2008
MSP430 Advanced Technical Conference

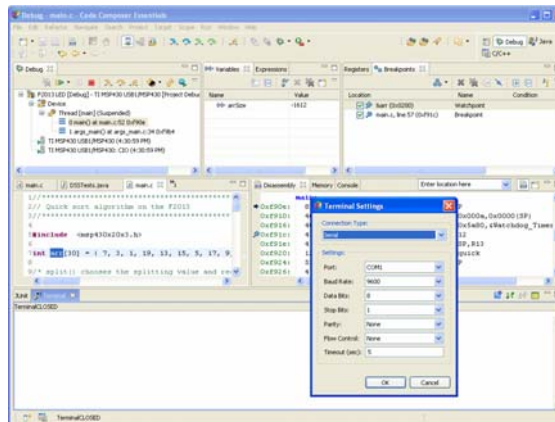
23



Terminal plug-in



- Supports COM port connection to your board



ATC 2008
MSP430 Advanced Technical Conference

24



Where to get additional plug-ins

- Eclipse.org community
 - Eclipse Plug-in Central
 - Browse through plug-ins
 - www.eclipseplugincentral.com
 - Older Eclipse Plug-in Site
 - www.eclipse.org/community/plugins.php
- If you still can't find what you want Google usually can



ATC 2008

MSP430 Advanced Technical Conference

25



Installing plug-ins

- As simple as copying files into your eclipse\plugins and eclipse\features folders
 - Typically distributed as a zip file
 - Shutdown Eclipse, unzip, start Eclipse
- CCE 3.0 uses Eclipse v3.2.0 and CDT v3.1
 - Plug-ins not designed for this version may not operate correctly
 - CDT plug-ins are unlikely to work at all due to modifications to it in CCE
- Do not update any core plug-ins
 - Overwriting these plug-ins will remove MSP430 modifications



ATC 2008

MSP430 Advanced Technical Conference

26



Software Development Organization

Plug-in example



- Demo
 - Installing Java support to CCE
 - Running DSS automated tests from within CCE

ATC 2008

MSP430 Advanced Technical Conference

27



Software Development Organization

Plug-in Development Environment



- Eclipse “Plug-in Development Environment” (PDE)
 - Used to create, test, debug, and deploy plug-ins
- Plug-ins are written in Java
 - “Java Development Toolkit” or JDT must be installed
 - available from <http://www.eclipse.org>

ATC 2008

MSP430 Advanced Technical Conference

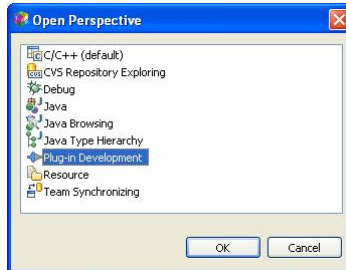
28



Software Development Organization

PDE perspective

- Window->Open Perspective->Other
 - Select Plug-in Development



ATC 2008

MSP430 Advanced Technical Conference

29



Software Development Organization

Plug-in development references

- Several different templates provided
- Tutorials available in the help and on the web
- A number of books available



ATC 2008

MSP430 Advanced Technical Conference

30



Workspaces

- What are they?
 - A workspace is a folder that contains information relevant to what you are working on
- A workspace contains
 - Projects
 - Either physically in the workspace or referenced by the workspace
 - Window and toolbar arrangements
 - Preference settings
- You can switch between multiple workspaces
- Automatically saved on exit
- Can be automatically loaded at startup



ATC 2008

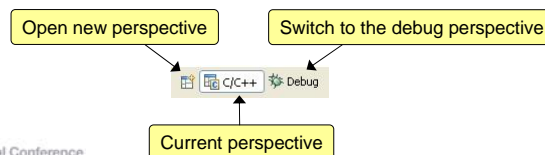
MSP430 Advanced Technical Conference

31



Perspectives

- Background:
 - Each Workbench window contains one or more perspectives
 - A perspective defines the initial set and layout of views
 - Each perspective is aimed at accomplishing a specific type of task
- As you work in the Workbench, you will probably switch perspectives frequently
- Customization
 - Perspectives control what appears in certain menus and toolbars
 - They define visible action sets, which you can change to customize a perspective
 - You can save a custom perspective that you build
- Default perspectives in CCE
 - C/C++: Editing and building
 - Debug: Debugging your programs
- Perspectives can be opened in the same window, or a new one



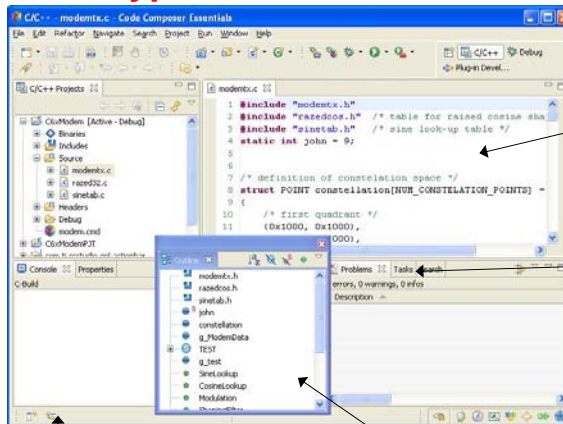
ATC 2008

MSP430 Advanced Technical Conference

32



Window Types



Editor:
Only editor windows are part of this group

Tab Group:
Several windows grouped together

Fast view: Hidden until you click on the button to restore them. Click on another window to hide

Detached:
Can float outside of the main window

ATC 2008

MSP430 Advanced Technical Conference

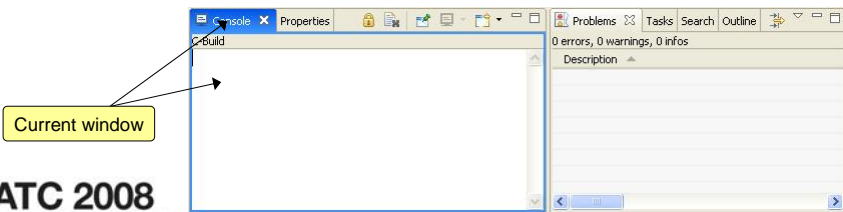
33



Windowing Tips



- Double-clicking a window's title bar will maximize it
 - Double-clicking again will restore it to its previous size
- Fast-view windows are great for windows used infrequently which require a lot of space
- The window that has focus is indicated by a blue border and heading



ATC 2008

MSP430 Advanced Technical Conference

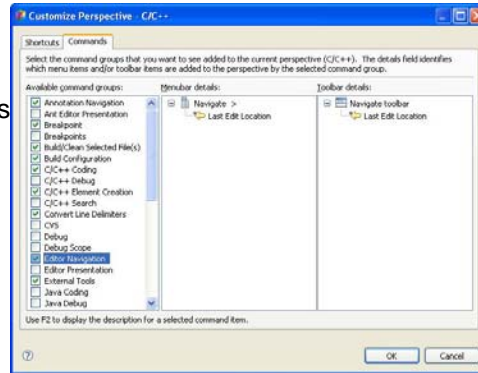
34



Customizing perspectives



- You can customize the menu items and toolbars in your perspective
- Commands
 - Controls menus & toolbars
- Shortcuts
 - Controls special sub-menus (new, show view...)



ATC 2008

MSP430 Advanced Technical Conference

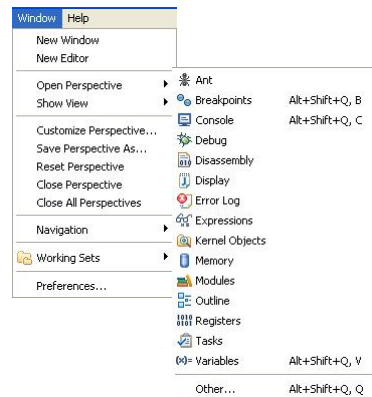
35



Accessing views



- To open a new view go to the Windows -> Show View Menu
 - Common views for the current perspective
 - Recently opened views
- To access views that are not listed select Other...



ATC 2008

MSP430 Advanced Technical Conference

36



Editor features

- Code Completion (ctrl + space)
 - Complete word
 - Auto-member information
 - Auto-parameter information
 - ...
- Navigation
 - Back/Forward buttons
 - Back to last edit button
 - Go to definition
 - Go to declaration
- Show line numbers
- Code Folding
 - Collapse functions



```

124
125 /* shift the delay line */
126 for(i = 0; i < SIZE_SHAPING_F)
127 {
128     delayLine[i] = delayLine[i
129 }
130
131 g_ModemData.
132
133 /* clear end
134 for( ; i < S
135 {
136     delayLin
137 }
138
139 /* add new s
140 for( i = SIZ
141 {
142     #if defined(PROD_C6X)
143     delayLine[i] = // (shortly
    
```

```

53 *****
54 #int SineLookup(int sample)
72
73 *****
    
```

ATC 2008

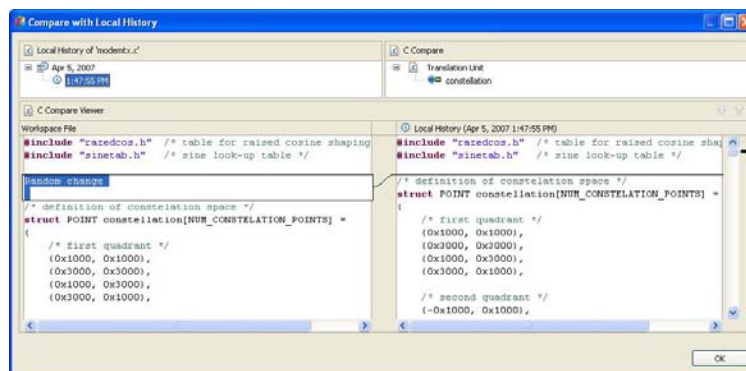
MSP430 Advanced Technical Conference

37



Local history

- Eclipse keeps a local history of source changes
- You can compare or replace your current source file with any previous saved version



ATC 2008

MSP430 Advanced Technical Conference

38



Breakpoint tips

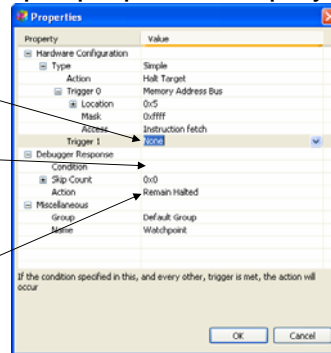


- Breakpoints are more than just the pre-made selections in the drop down
 - Right clicking on one can bring up a properties display

And additional triggers together

Define a condition on which to halt

Perform actions when it's hit such as saving memory to a file, or writing memory from a file



ATC 2008

MSP430 Advanced Technical Conference

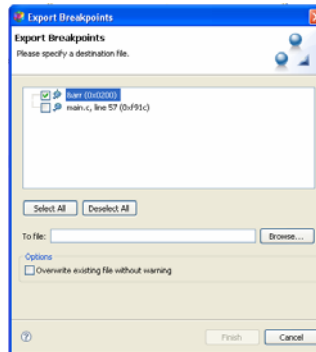
39



Breakpoint tips



- Have a set of breakpoints for a specific purpose?
 - You can save one or more breakpoints to a file to use later in another workspace, or even share with other users



ATC 2008

MSP430 Advanced Technical Conference

40



Detect invalid branch



- Fill memory with 0x4343
- Load code with the option *Retain Unchanged Memory*
 - 0x4343 is the opcode used for software breakpoints
 - If the program branches to invalid memory, it will halt

```
MEMORY
{
  SFR          : origin = 0x0000, length = 0x0010
  PERIPHERALS_8BIT : origin = 0x0010, length = 0x00F0
  PERIPHERALS_16BIT : origin = 0x0100, length = 0x0100
  RAM          : origin = 0x0200, length = 0x0080
  INFOA       : origin = 0x10C0, length = 0x0040
  INFOB       : origin = 0x1080, length = 0x0040
  INFOC       : origin = 0x1040, length = 0x0040
  INFOD       : origin = 0x1000, length = 0x0040
  FLASH       : origin = 0xF800, length = 0x07E0
  INT00       : origin = 0xFFE0, length = 0x0002
}

js: > session.memory.fill( 0x200, 0, 0x40, 0x4343)
js: > session.memory.fill( 0xF800, 0, 0x7e0/2, 0x4343)
js: >
```

- Trace, if available, can detect where the faulty branch occurred

ATC 2008

MSP430 Advanced Technical Conference

41



Additional References



DSS: Target setup



- DSS used from the Scripting Console uses the active debug session automatically
- DSS used from the command line requires you to indicate what the target is setup

Choose the Setup tab

Select TI Debug Settings

Click Export

Save to <install>\DebugServer\bin\win32\SystemSetup.xml

ATC 2008

MSP430 Advanced Technical Conference

43



DSS: Generated xml-based log



```
<?xml version="1.0" encoding="windows-1252" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="SimpleTransform.xsl"?>
<log>
<record>
<date>2007-05-02T15:30:15</date>
<millis>1178134215917</millis>
<sequence>0</sequence>
<logger>com.ti</logger>
<level>FINER</level>
<class>com.ti.ccstudio.scripting.environment.ScriptingEnvironment</class>
<method>traceSetConsoleLevel</method>
<thread>10</thread>
<message>RETURN</message>
</record>
<record>
<date>2007-05-02T15:30:15</date>
<millis>1178134215917</millis>
<sequence>1</sequence>
<logger>com.ti</logger>
<level>FINER</level>
<class>com.ti.ccstudio.scripting.environment.ScriptingEnvironment</class>
<method>getServer</method>
<thread>10</thread>
<message>ENTRY sServerName: LegacySetupServer.1</message>
</record>
```

ATC 2008

MSP430 Advanced Technical Conference

44



DSS: What about Perl?



- Is it possible?
 - YES
- Is it complicated?
 - NO
- Required Components:
 - Perl 5.8.3 or higher
 - <http://www.activestate.com/Products/ActivePerl/?tn=1>
 - Java 2 SDK (for javac)
 - <https://java.sun.com/javase/downloads/index.jsp>
 - Inline::Java
 - <http://theoryx5.uwinnipeg.ca/ppms/Inline-Java.ppd>

ATC 2008

MSP430 Advanced Technical Conference

45



DSS: Inline::Java for Perl



- What?
 - DSS APIs are in Java. Inline::Java perl module lets us access Java classes from perl
 - **Perl Inline::Java method interfaces directly to the supported DSS Java APIs**
 - A Java compiler is launched and the Java code is compiled.
 - The Java classes and methods are available to the Perl program as if they had been written in Perl.
 - The process of interrogating the Java classes for public methods occurs the first time you run your Java code. The namespace is cached, and subsequent calls use the cached version
- Why?
 - Perl is one of the most popular scripting languages. Huge availability of modules on [CPAN](http://cpan.org) is big strength.
- When?
 - Perl [Inline](http://cpan.org) module has been around for 7+ years, initially developed to leverage C code from within a perl environment. In the last ~5 years more language support has been added. Inline::Java was initially developed in 2001

ATC 2008

MSP430 Advanced Technical Conference

46



DSS: DSS_SCRIPTING.pm



- Perl module that hides a bunch of generic stuff that every .pl script would otherwise have to know/learn
 - To use: -
 - **# import module abstracting the DSS class details**
 - **use DSS_SCRIPTING;**

 - **# call the DSS class constructor(s) we need**
 - **my \$dss = new DSS_SCRIPTING();**
- Details (purpose of module is to hide these from user)
 - We tell Inline::Java to explicitly 'STUDY' main DSS class (ScriptingEnvironment) so we can use it from perl
 - We turn on AUTOSTUDY which makes Inline::Java automatically study unknown classes as it encounters them
 - We proxy Java's importPackage() by assigning a new namespace TraceLevel to the fully scoped package name so we can then use the shorter namespace \$TraceLevel::ALL instead of \$DSS_SCRIPTING::com::ti::ccstudio::scripting::environment::TraceLevel::ALL
 - Finally, we enable access to the ScriptingEnvironment Java class constructor

ATC 2008

MSP430 Advanced Technical Conference

47



DSS: Perl – Simple Example



- Example
 - Simplistic example to show that Perl works with DSS
- What does the Script do?
 - Starts the debug server
 - Connects to the first target
 - Loads the program passed as a parameter
 - Enables analysis events
 - Sets a breakpoint at main
 - Runs the target
 - Checks where it is halted
 - Runs the target
 - Disables the events
 - Exits
- Executing the script
 - **perl dss_simple.pl [outfile] [setup config file]**

ATC 2008

MSP430 Advanced Technical Conference

48



DSS: Example: automated testing



- A common use for DSS is to automate testing
- If you don't want to write your own framework, use something freely available such as junit
 - JUnit requires raw java code, not javascript
 - You'll need a .bat file to setup the path and classpath correctly first
 - Simply modify rhino.bat to call your java object

```
REM include JUNIT jars in the path
set JUNIT_JARS=D:\testing\junit4.3\junit-4.3.jar

REM include the test files in the classpath
set DSS_TESTS_DIR=C:\DOCUME~1\ad792176\DSS_Example

REM Launch the b.SS test file, not rhino
set DSS_TESTS=DSTest.m

REM *****
REM Launch Rhino script engine. Import the scripting package.
java.exe -DXPCOM.RUNTIME="%DEBUGSERVER%\win32" -cp %RHINO_JARS%;%SCRIPTING_JARS%;%DSS_TESTS_DIR%;%JUNIT_JARS% %DSS_TESTS% %1 %2 %3 %4 %5
```

ATC 2008

MSP430 Advanced Technical Conference

49



DSS: Example: automated testing



- Create a junit testing object and compile it
 - Create a function with @BeforeClass to initialize a DSS session
 - Create a function with @AfterClass to terminate the session
 - Create functions with @Before and @After to setup the target and clean up after any test function
 - Then finally use @Test to write a bunch of testing functions
 - Validate algorithms are correct with different data sets
 - Validate application performance with the profile functions
- Demo

ATC 2008

MSP430 Advanced Technical Conference

50



DSS: References



- Javascript References
 - Rhino – Javascript for Java: <http://www.mozilla.org/rhino/>
 - W3Schools – Javascript Tutorial: <http://www.w3schools.com/js/default.asp>
- XSLT (XML Stylesheet Transforms) References.
 - If the format of the log file isn't exactly what you need.
 - XSLT Tutorial: http://www.w3schools.com/xsl/xsl_intro.asp
 - Xalan – XSLT Processor: <http://xml.apache.org/xalan-j/>
- JUnit References
 - JUnit home page: <http://www.junit.org/>
 - JUnit FAQ: <http://junit.sourceforge.net/doc/faq/faq.htm>

ATC 2008

MSP430 Advanced Technical Conference

51



Installing java support



- CCE does not ship with java support, but it can be easily added
 - Go to <http://www.eclipse.org> and click on *PROJECTS* near the top
 - Click on *Show me all the projects*
 - Click on *Eclipse Project*
 - Click *Download* in the corner
 - From here, there should be a link to archived builds of earlier versions
 - Click on the link for version 3.2
 - Download eclipse-SDK-3.2-win32.zip from this page
 - Extract all files on top of the CCE 3.2 install, but do not overwrite existing files
 - Existing files contain TI specific changes necessary to debug TI devices
- Now that CCE supports java, you can use it to develop and run DSS based JUnit tests

ATC 2008

MSP430 Advanced Technical Conference

52



Creating a DSS JUnit project



- Switch to the Java perspective
- Right click on the project view and select project and give it a name
- Right click on the new project and add a new class
- Right click on the new project and select *Build Path->Add External Libraries...*
 - Select the DSS jars (all the jars in DebugServer\scripting\lib)
 - Download JUnit from the web (<http://www.junit.org/>) and add it's jar

ATC 2008

MSP430 Advanced Technical Conference

53



Creating a DSS JUnit test



- You'll need to include the necessary imports, and then just define startup, shutdown routines

```

import static org.junit.Assert.*;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import com.ti.coustudio.scripting.environment.ScriptingEnvironment;
import com.ti.debug.engine.scripting.DebugServer;
import com.ti.debug.engine.scripting.DebugSession;
import com.ti.debug.engine.scripting.ProfileServer;
import com.ti.debug.engine.scripting.ProfileSession;

public class DSSTests {
    // This allows the use of earlier junit's to be used.
    public static junit.framework.Test suite() {
        return new junit.framework.JUnit4TestAdapter(DSSTests.class);
    }
    // Main function which runs junit with this object as the class to test
    public static void main(String args[]) {
        org.junit.runner.RunWith(org.junit.runners.JUnit4.class);
    }

    private static ScriptingEnvironment env;
    private static DebugSession session;

    @BeforeClass
    public static void oneTimeSetup() throws Exception {
        // Get a new session
        env = new ScriptingEnvironment();
        DebugServer server = (DebugServer) env.getServer("debugserver.1");
        session = server.openSession();

        // Connect
        session.target.connect();

        // Called after running every test in this class
    }

    @AfterClass
    public static void oneTimeTearDown() throws Exception {
        session.terminate();
    }

    // Called before each test in this class
    @Before
    public void setUp() throws Exception {
        // Load a program
        session.memory.loadProgram("C:\\Documents and Settings\\40792178\\worksp
    }

    // Called after each test in this class
    @After
    public void tearDown() throws Exception {
        // Clear any breakpoints used by the test
        session.breakpoint.removeAll();
    }
}

```

ATC 2008

MSP430 Advanced Technical Conference

54



Creating a DSS JUnit test



- Then just define test routines

```

@Test
public void verifyExecution() throws Exception {
    // Run to the end of main
    session.breakpoint.add("main.c", 58);
    session.target.run();

    // Verify that the elements in the array were sorted
    for( int i = 0; i < 10; ++i )
    {
        long value1 = session.expression.evaluate( "arr[" + i + "]" );
        long value2 = session.expression.evaluate( "arr[" + (i + 1) + "]" );
        assertTrue( value1 <= value2 );
    }
}

@Test
public void profileProgram() throws Exception {
    // Verify that the program completes in a minimum amount of cycles
    ProfileServer ps = (ProfileServer) ( msw.getServer( "ProfileServer.1" ) );
    ProfileSession profileSession = ps.openSession( session );
    session.breakpoint.add("main.c", 58);
    long time = profileSession.runBenchmark();
    System.out.println( "quicksort took " + time + " cycles" );
    assertTrue( time < 2250 );
}

```

ATC 2008

MSP430 Advanced Technical Conference

55



Running a DSS JUnit test



- Right click on the test class and choose *Debug As->JUnit Test* or *Run As->JUnit Test*
- If you get an error about XPCOM.RUNTIME not being defined
 - Right click, and select *Debug As->Debug...*
 - Select *Arguments* and under *VM Arguments* enter XPCOM.RUNTIME=<install>\DebugServer\win32
 - Spaces in the path won't work, so use the ~ filenames: for instance
 DXPCOM.RUNTIME=C:\PROGRA~1\TEXAS~1\CESSE~1\DebugServer\win32\
 - Apply and then you can debug this normally

ATC 2008

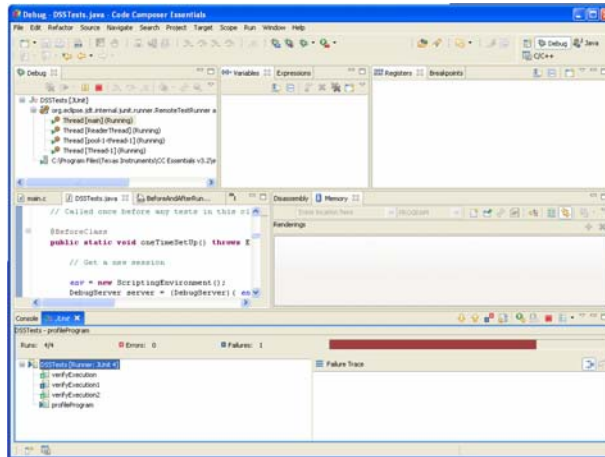
MSP430 Advanced Technical Conference

56



Running a DSS JUnit test

- Results will appear in the JUnit tab



ATC 2008

MSP430 Advanced Technical Conference

57



Creating a plugin

- Objectives
 - Create a plug-in that adds a menu item and toolbar button to CCS-Eclipse
 - Clicking the button will execute a GEL expression
 - Show that there are good tools for creating plug-ins



ATC 2008

MSP430 Advanced Technical Conference

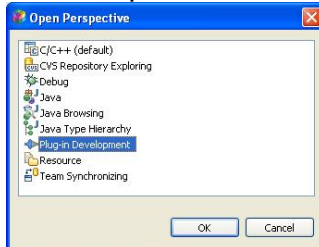
58





Step 1: Open the PDE perspective

- Window->Open Perspective->Other
 - Select Plug-in Development



- Window arrangement will change and you should see “Plug-in Development” in the perspective indicator

ATC 2008

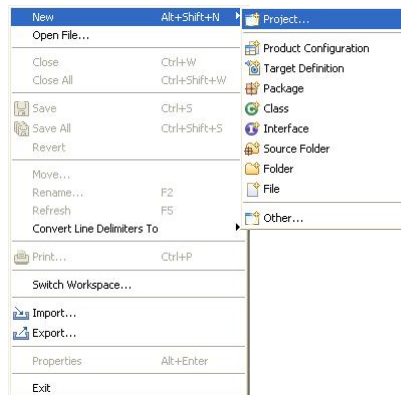
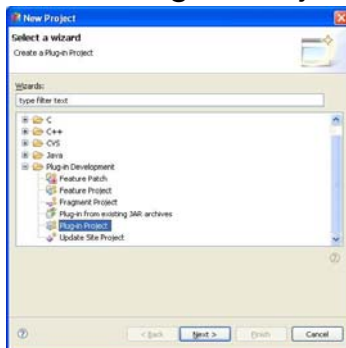
MSP430 Advanced Technical Conference

59



Step 2: Start a new project

- From the File menu select New->Project
- Select Plug-in Project



ATC 2008

MSP430 Advanced Technical Conference

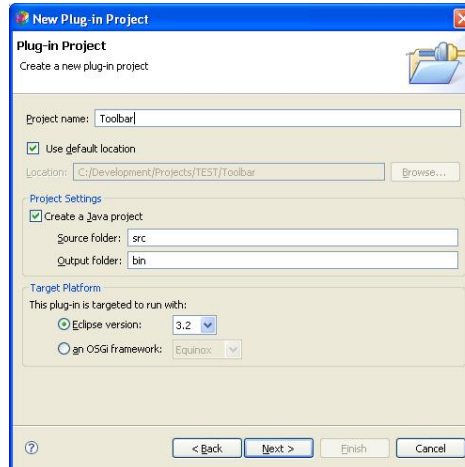
60



Software Development Organization

Step 3: Name the project

- Name your project “Toolbar”
- Will automatically create folders for source and binaries
- Select 3.2 in the Eclipse version
- Click “Next”



ATC 2008

MSP430 Advanced Technical Conference

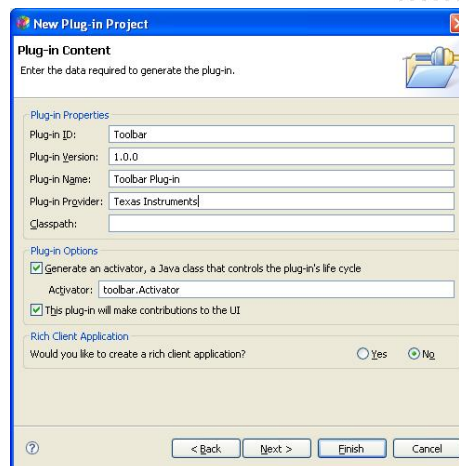
61



Software Development Organization

Step 4: Plug-in properties

- Fill in the Plug-in provider name – i.e. “Texas Instruments”
- Click “Next”



ATC 2008

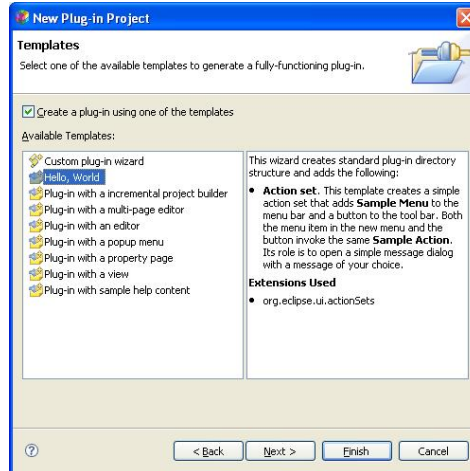
MSP430 Advanced Technical Conference

62



Step 5: Templates selection

- Select the “Hello World” template as it adds the menu and button we want
- Click “Next”



ATC 2008

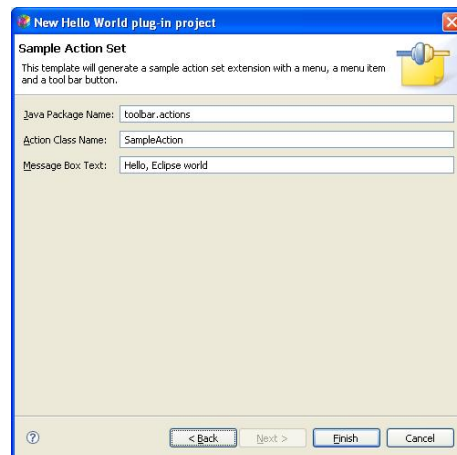
MSP430 Advanced Technical Conference

63



Step 6: Message box text

- Change the message to whatever you want
- Click “Finish”




ATC 2008

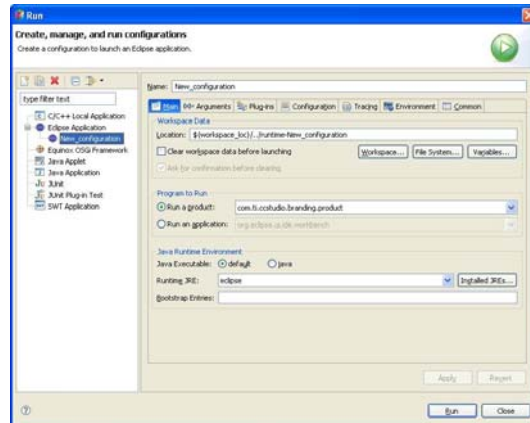
MSP430 Advanced Technical Conference

64



Step 7: Create a run configuration

- Click the Run button 
- Double-click on Eclipse Application to create a new configuration
- Click “Run”





ATC 2008

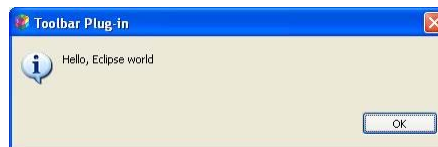
MSP430 Advanced Technical Conference

65



Step 8: Test the plug-in

- When CCS-Eclipse starts click the yellow arrow  to go to the workbench
- Click on the purple Eclipse button  on the toolbar to test your plug-in



- Click “Ok”
- Exit this instance of CCS-Eclipse

ATC 2008

MSP430 Advanced Technical Conference

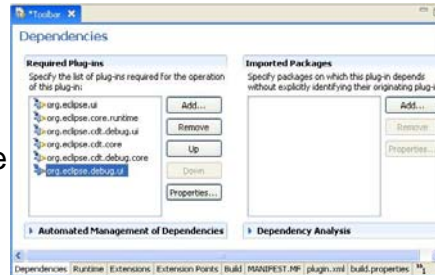
66





Step 9: Dependent plug-ins

- We are going to be using APIs from other components so we have to specify them
- Click on the “Toolbar” source file
- Click on the “Dependencies” tab at the bottom of the window
- Add
 - org.eclipse.cdt.debug.ui
 - org.eclipse.cdt.core
 - org.eclipse.cdt.debug.core
 - org.eclipse.debug.ui
- Save the file



ATC 2008

MSP430 Advanced Technical Conference

67



Step 10: Add required imports

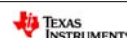
- In the import statement area at the top of SampleAction.java add these import statements as we will be using these classes

```
import org.eclipse.cdt.debug.core.cdi.CDIException;  
import org.eclipse.cdt.debug.core.cdi.model.ICDIStackFrame;  
import org.eclipse.debug.ui.DebugUITools;
```

ATC 2008

MSP430 Advanced Technical Conference

68



Step 11: Change to evaluate expression

- Double click on SampleAction.java in the Package Explorer to open the file
- In the “run” function replace:

```
MessageDialog.openInformation(window.getShell(),  
    "Toolbar Plug-in",  
    "Hello, Eclipse world");
```

- So that it looks like this:

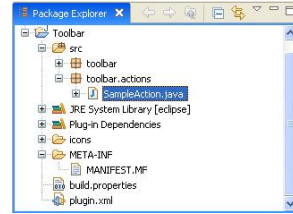
```
public void run(IAction action) {  
    // Get the current debug context so that we know which context to apply the GEL call to  
    ICDIStackFrame frame = (ICDIStackFrame)DebugUITools.getDebugContext().getAdapter(ICDIStackFrame.class);  
    // Call GEL  
    try {  
        frame.getTarget().evaluateExpressionToString(frame, "GEL_Run()");  
    } catch (CDIException e) {  
        e.printStackTrace();  
    }  
}
```

- Save

ATC 2008

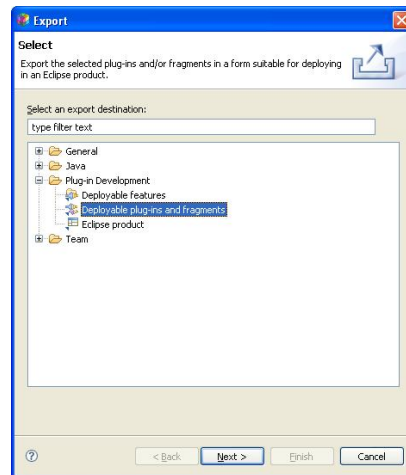
MSP430 Advanced Technical Conference

69



Step 12: Export

- You want to be able to give your plug-in to other people
- File->Export
- Select “Deployable plug-ins and fragments”



ATC 2008

MSP430 Advanced Technical Conference

70



Software Development Organization

Step 13: Export location

- Specify the location to export the plug-in to
 - Specify the \eclipse folder of your CCS-Eclipse installation
- Click “Finish”



ATC 2008

MSP430 Advanced Technical Conference



Software Development Organization

Step 14: Test in CCE

- Close CCE
- Start CCE
- Start the debugger
- Load a program
- Click on your button



ATC 2008

MSP430 Advanced Technical Conference

72

