



Hands-On Workshop: Mastering **Kinetis Design Studio** IDE - Advanced FTF-DES-F1149

Erich Styger | MCU Software and Tools Marketing Manager
Mark Ruthenbeck | Applications Engineering

JUNE 2015



External Use



Agenda

- Introduction to Kinetis Design Studio IDE
- Labs:
 - Version Control
 - Build Automation
 - Debug Automation
 - Coverage
- Wrap-up



Overview

- Precondition: Familiar with KDS/Eclipse
- Goal: Exploring advanced possibilities
 - Version Control → Command line build & debug → Coverage

[1]
Version Control
Using Project with VCS/Git

[3]
Debug Automation
Debugging with gdb

[2]
Build Automation
Building/Generating Code

[4]
Coverage
Gcov with Eclipse

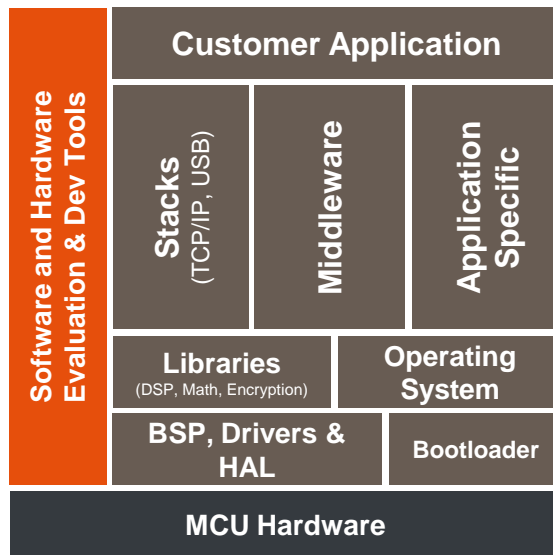
Kinetis Design Studio IDE



No-cost integrated development environment (IDE) for Kinetis MCUs



Eclipse and GCC-based IDE for C/C++ editing, compiling and debugging



Product Features

- A free of charge and unlimited IDE for Kinetis MCUs
- A basic IDE that offers robust editing, compiling and debugging
- Based on Eclipse, GCC, GDB and other open-source technologies
- Includes Processor Expert with Kinetis SDK integration
 - Supports all existing Kinetis devices via PEx and new project wizard
 - All new Kinetis devices will also feature the Kinetis SDK with Processor Expert configurability
 - Code Generation for GNU, IAR and Keil
- Host operating systems:
 - Windows 7/8 (32 and 64-bit)
 - Linux (Ubuntu, Redhat, Centos) (64bit)
 - Mac OS X and Segger J-Link
- Support for SEGGER, P&E and Open SDA/CMSIS-DAP debugger targets
- Support for Eclipse plug-ins including RTOS-awareness (i.e. MQX, FreeRTOS)

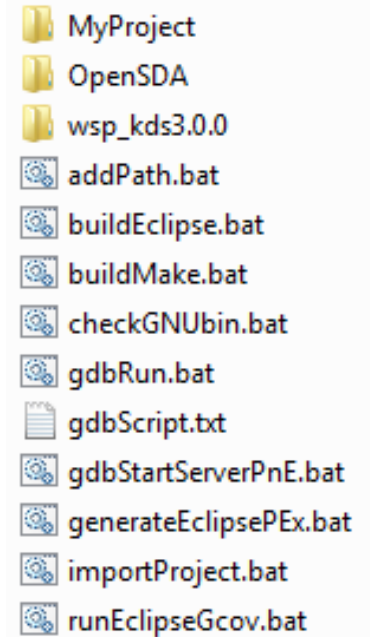
Learn more at: www.freescale.com/KDS

Community: <https://community.freescale.com/community/kinetis-design-studio>



Lab Notes

- Lab Computer Password: **CodeWarrior1**
- Lab Files in **C:\FTF\F1149**
- MyProject
 - KDS Project used in Lab
- Batch/script files
 - Simple batch/command files used during the labs
 - Run from cmd.exe (DOS Shell), do NOT just double click on the files!



Please leave the boards/cables/notebooks in the room. We need them for other classes!

Version Control

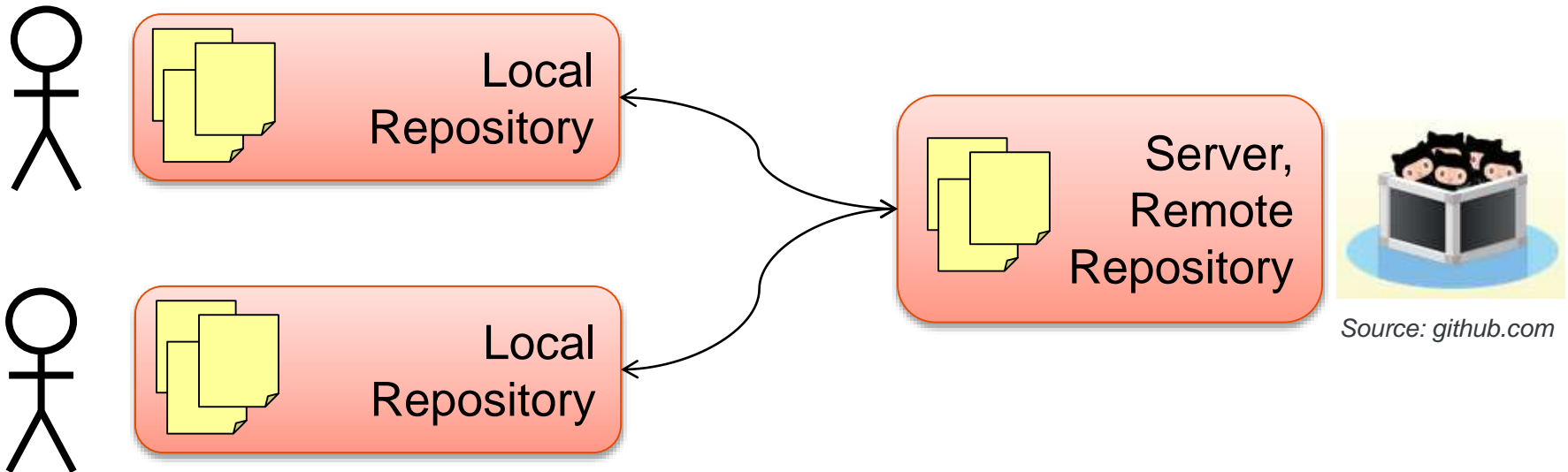


Why Version Control?

- Collaboration between multiple developers (same project and files)
- Backup and Restore, Synchronization
- Short-term undo, Long-term undo
- Track Changes, Track Ownership
- Sandbox, Branching, Merging

Typical (Distributed) Version Control System

- Developers work on copy of files (local repository, 'clones')
- Server is managing files in repository (e.g. Git, GitHub)
- Users are connected to repositories with connectors (e.g. eGit in Eclipse)



Source: eclipse.org

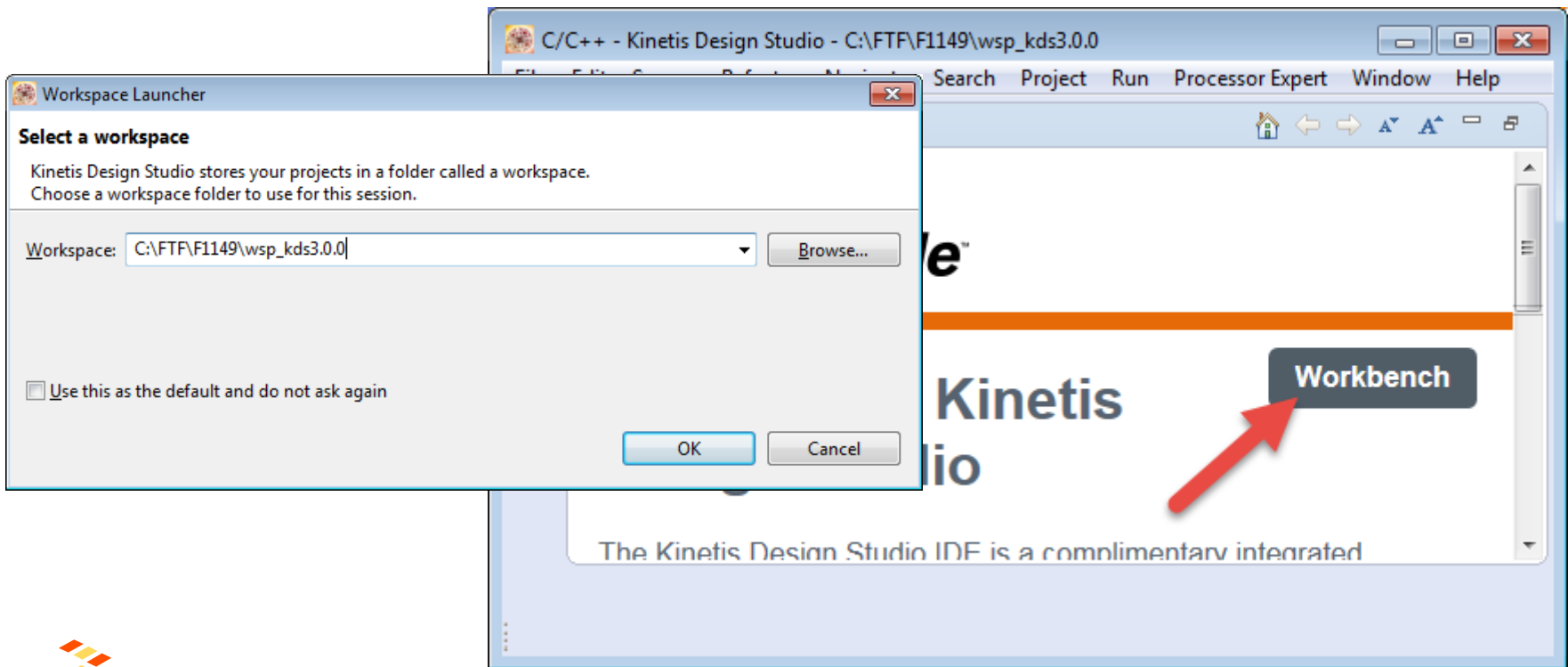
Lab Outline

- Start with a new/empty workspace in Kinetis Design Studio
- Import lab project 'MyProject' into workspace
- Create a new LOCAL git repository
 - no connection to network/GitHub
- Put MyProject under Version Control (git)
 - Add, commit, history, diff, ...
- Note: in this lab, we will use a 'personal'/local (not shared on the network) repository



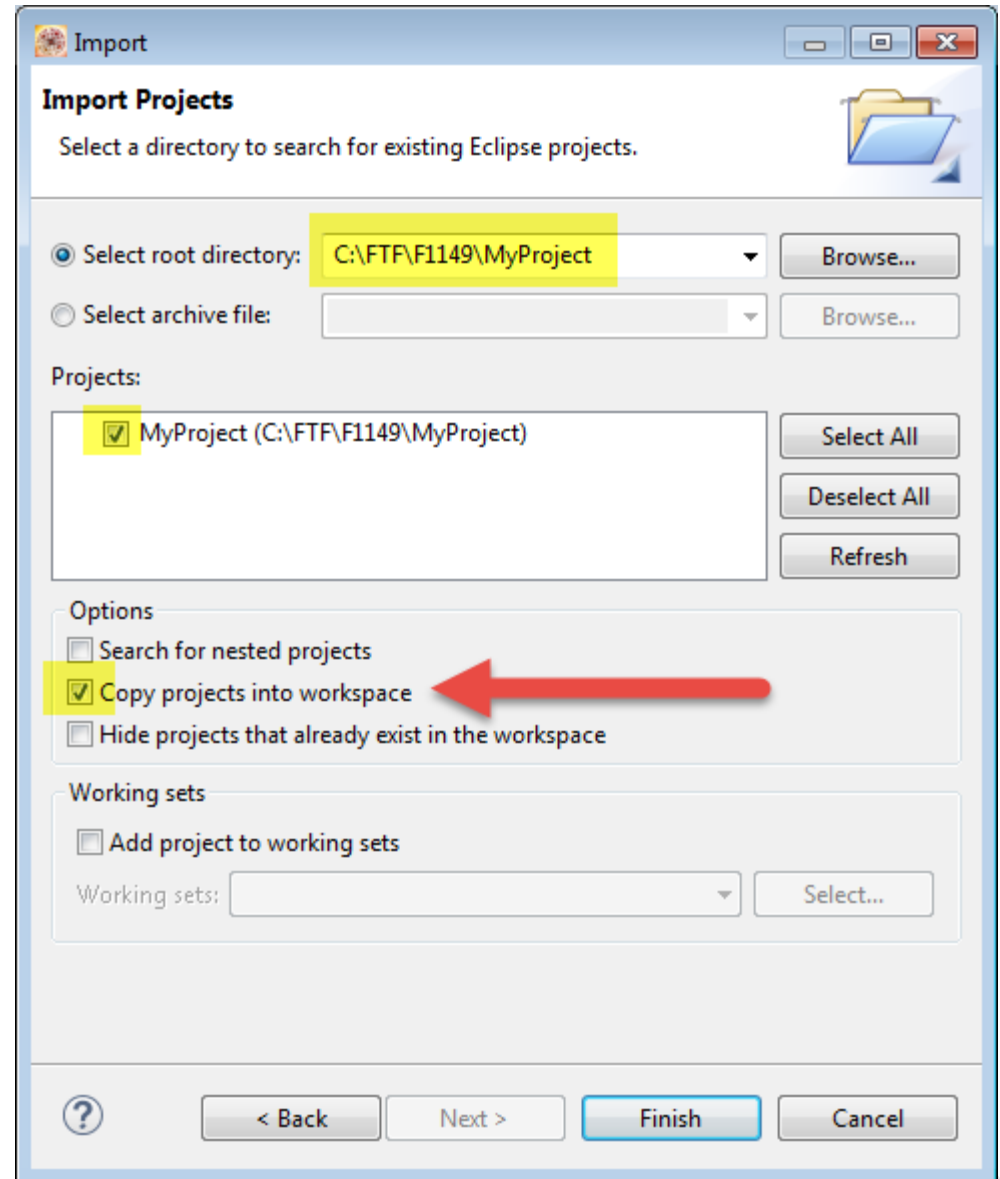
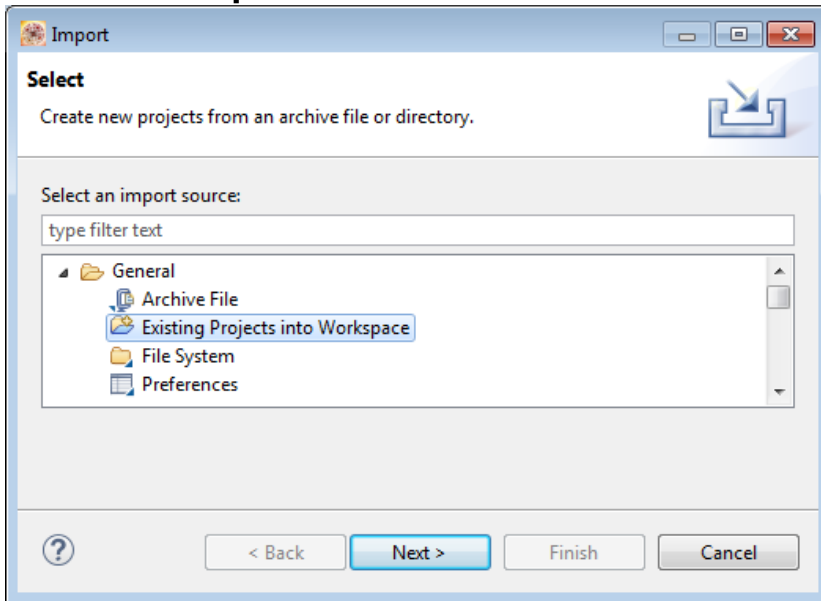
Starting Kinetis Design Studio

- Start Kinetis Design Studio
- Use following workspace
 - C:\FTF\F1149\wsp_kds3.0.0
- Switch to Workbench



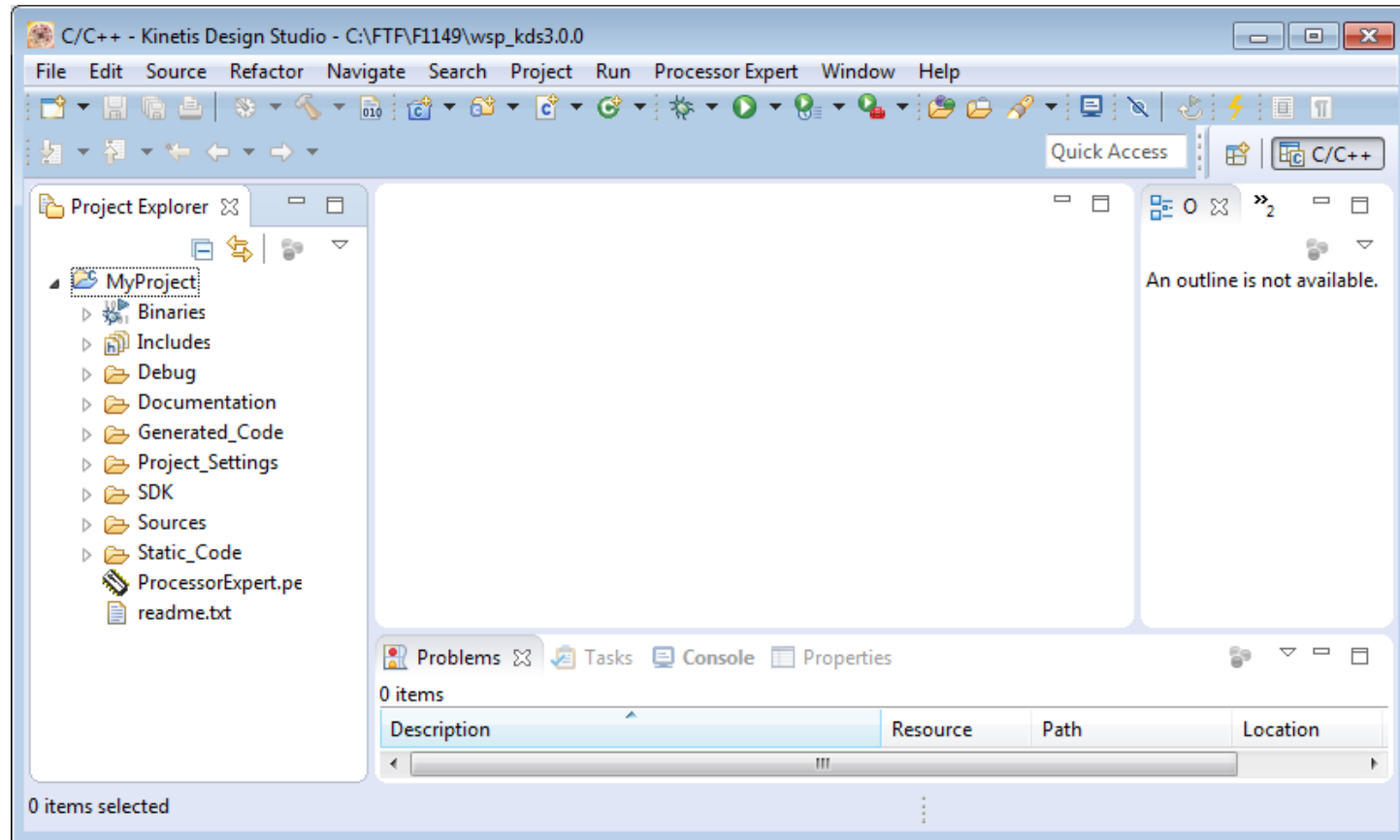
Import Existing Project

- Menu *File* > *Import* > *General* > *Existing Projects into Workspace*
- C:\FTF\F1149\MyProject
- **Copy** projects into workspace



Project Imported into Workspace

- Project copied into workspace folder
- Project shown in Project Explorer

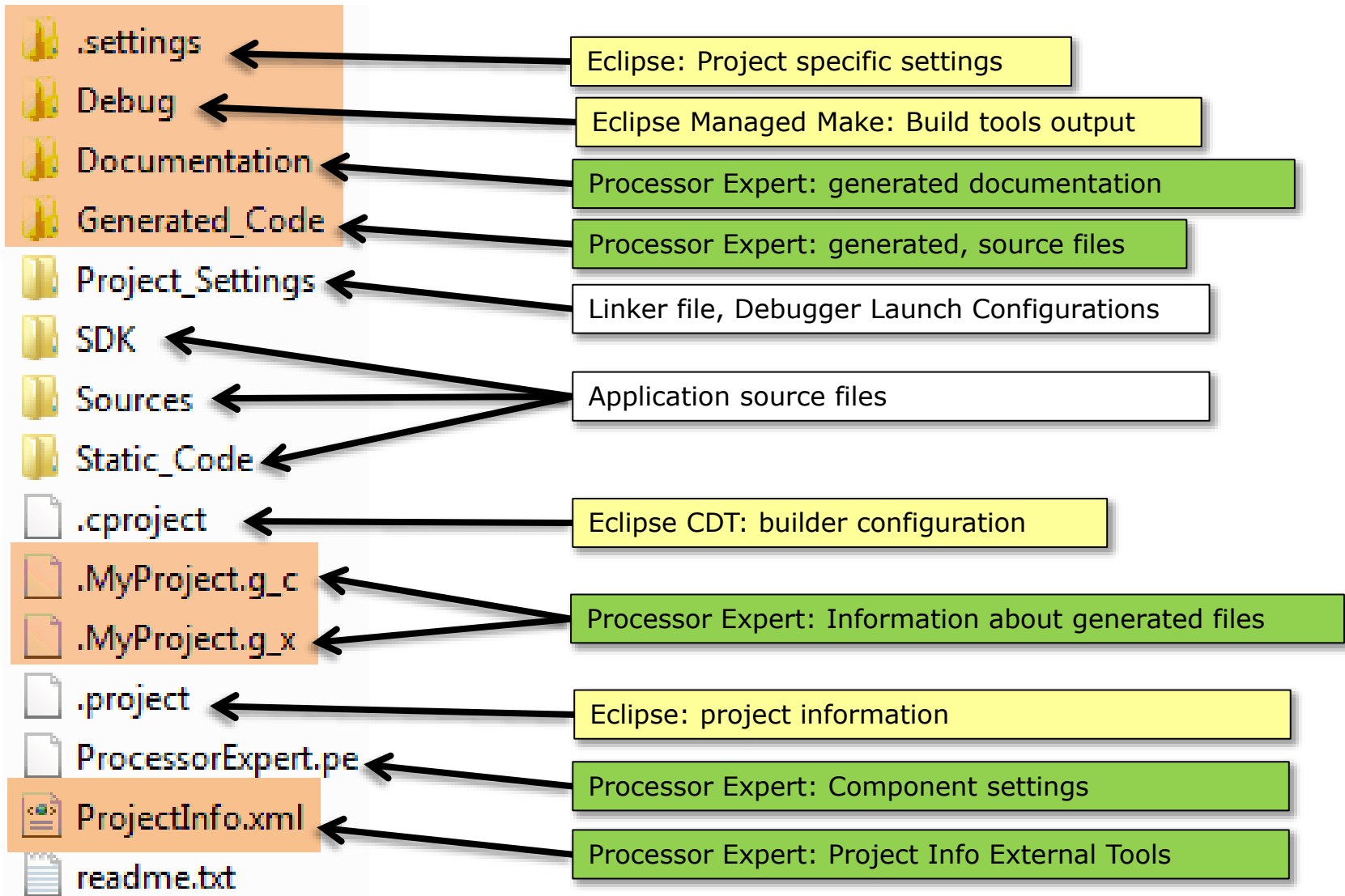


Version Control Rule #1

**DO NOT PUT DERIVED/GENERATED FILES
INTO A VERSION CONTROL SYSTEM**

- Files will be generated/created by build
- Putting them into the system only will waste space and create conflicts!
- Need to understand Eclipse & Processor Expert project files

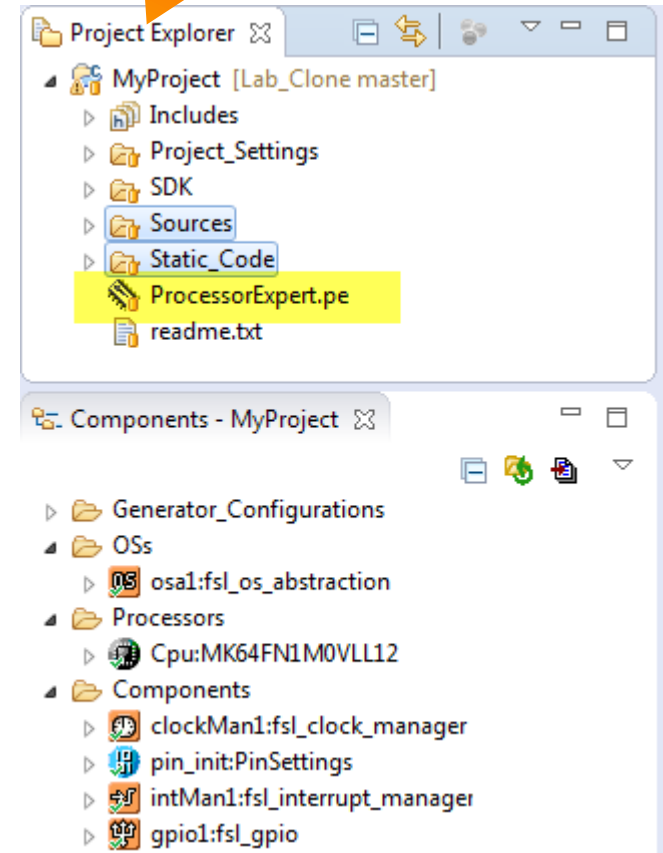
Understanding Project Files and Folders



CAUTION: ProcessorExpert.pe Files

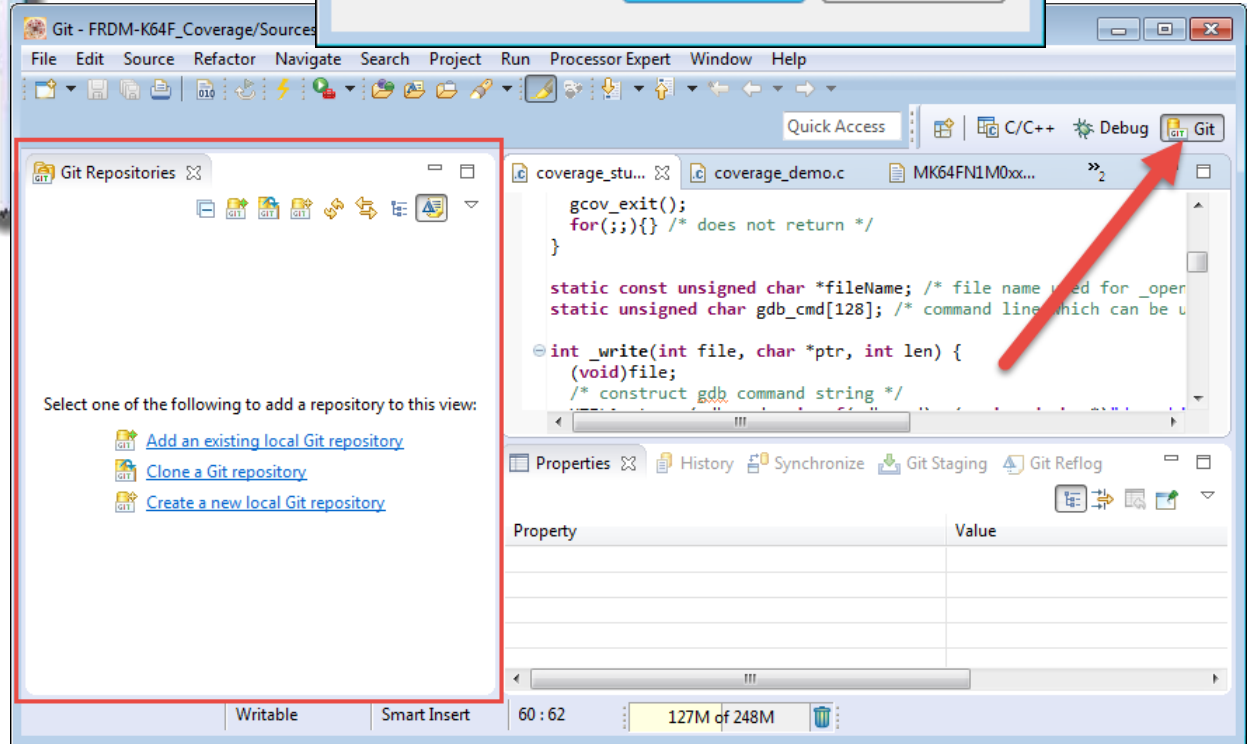
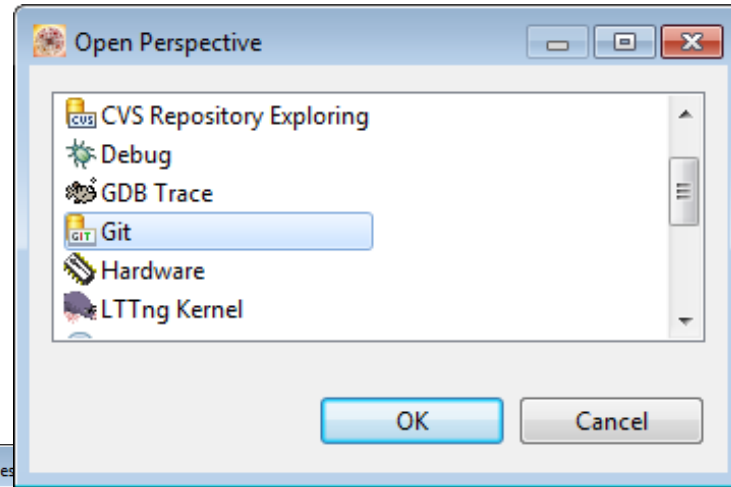
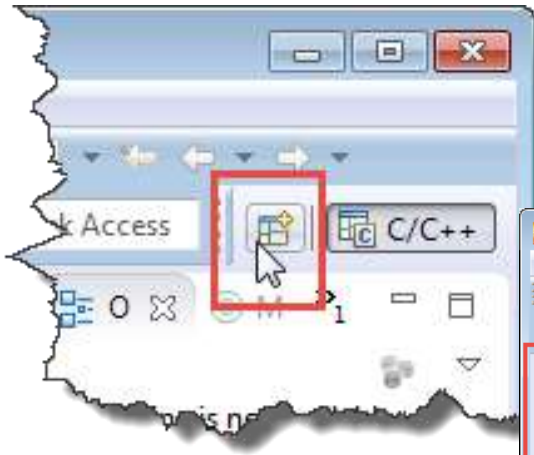
- Problem with sharing .pe file with **multiple users**
- XML file with all the Processor Expert settings
 - File can get really huge (several MBytes)
 - Contains components, component settings, etc
- Not ideal for any version control system
 - One user makes a change
 - Likely conflicting with changes from another user
 - Difficult to merge (XML structure)
- Pragmatic approach ☹
 - Handling like a 'binary' file
 - Only one developer does a change, commit
 - All others pull the change

Danger Zone!



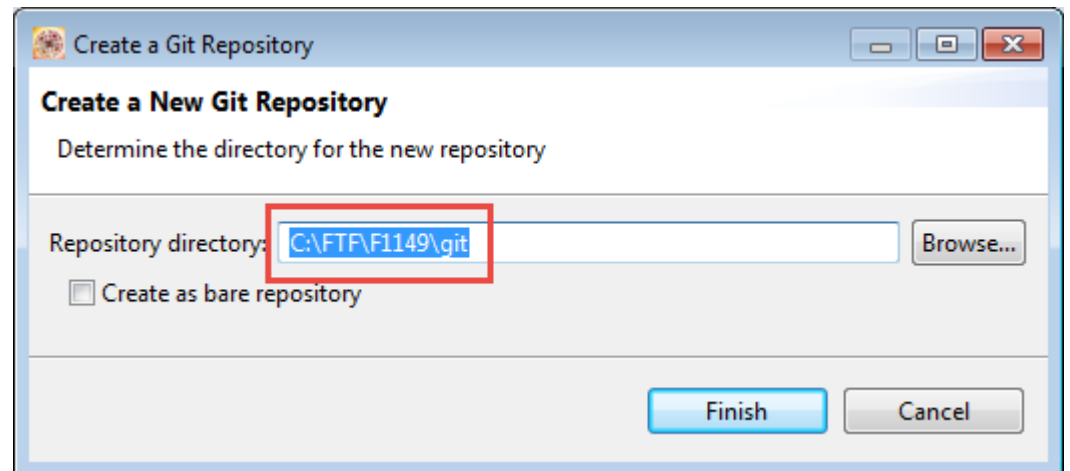
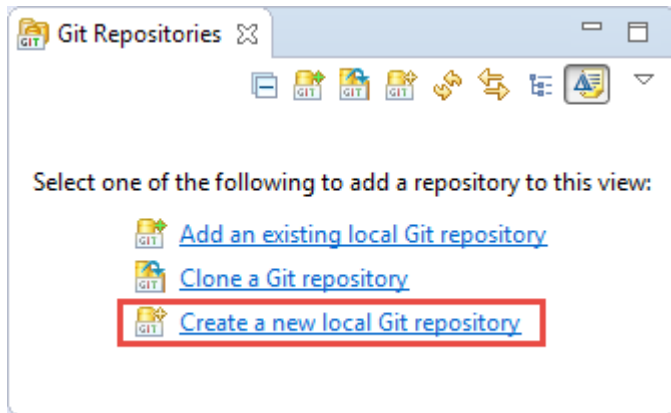
Git Perspective

- Switch to 'Git' Perspective



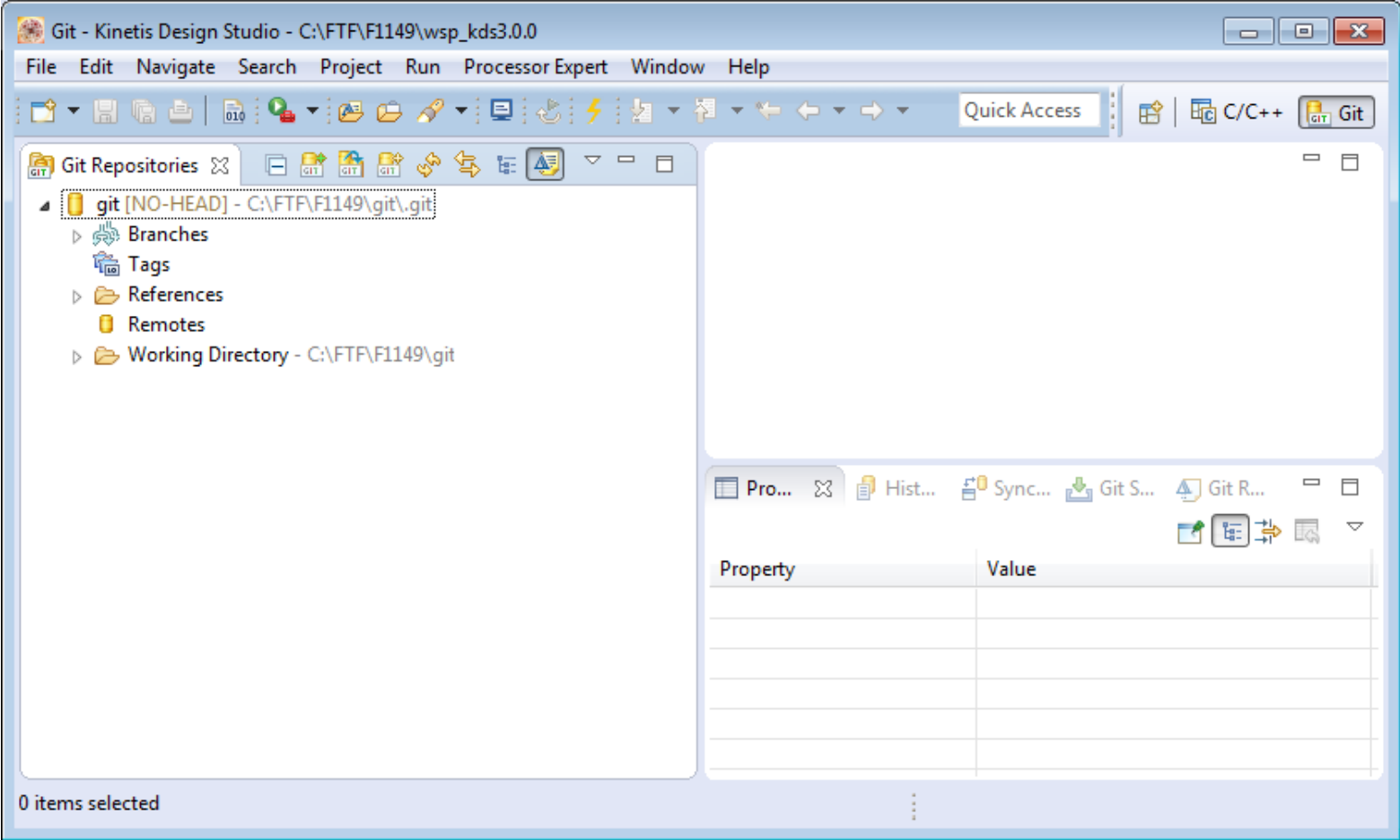
Create a New Git Repository

- Create new local Git repository
- Specify repository location/directory
 - C:\FTF\F1149\git
- Note: we are NOT creating a 'bare' repository. A 'bare' one would be used to share it with multiple developers



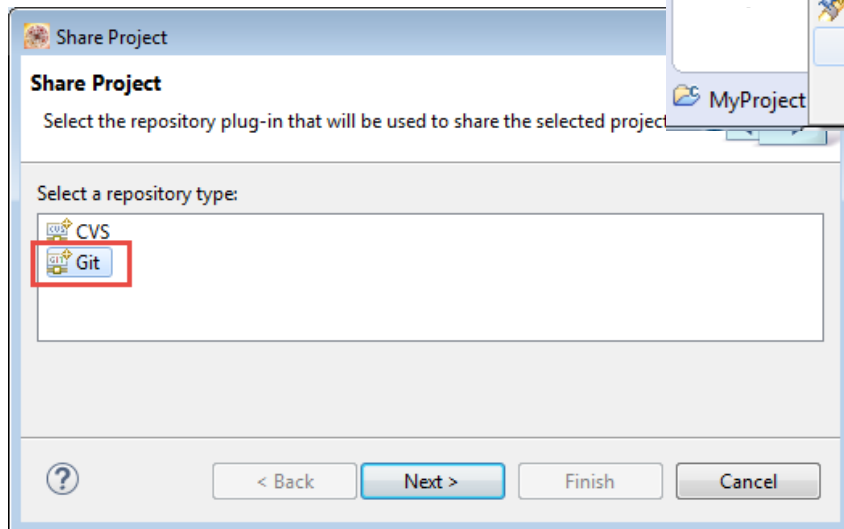
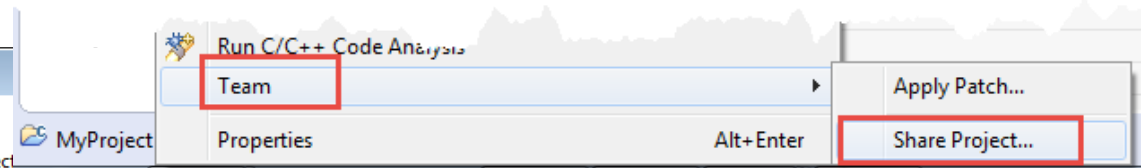
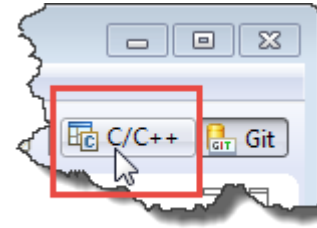
Git Repository

- New repository added to view



Adding Project to Repository

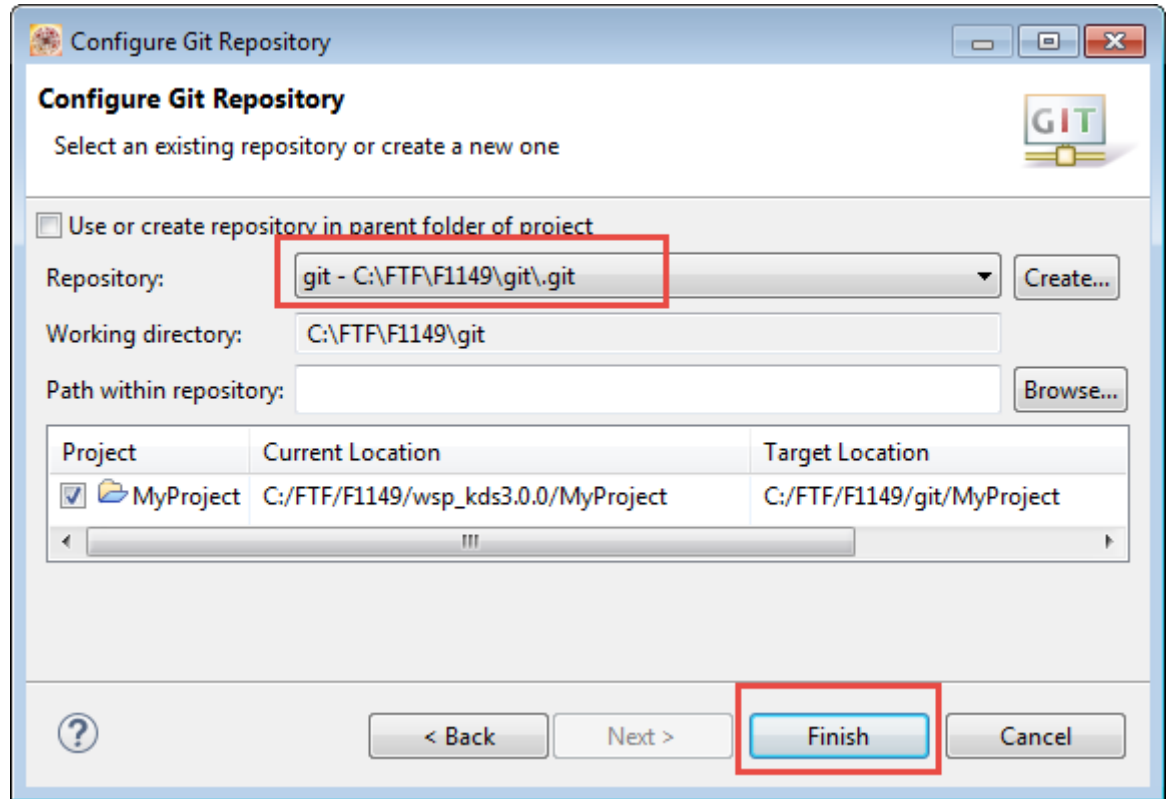
- Switch to C/C++ Perspective
- Context menu: *Team* > *Share Project*
- Select 'Git'
- Press 'Next'



Configure Git Repository

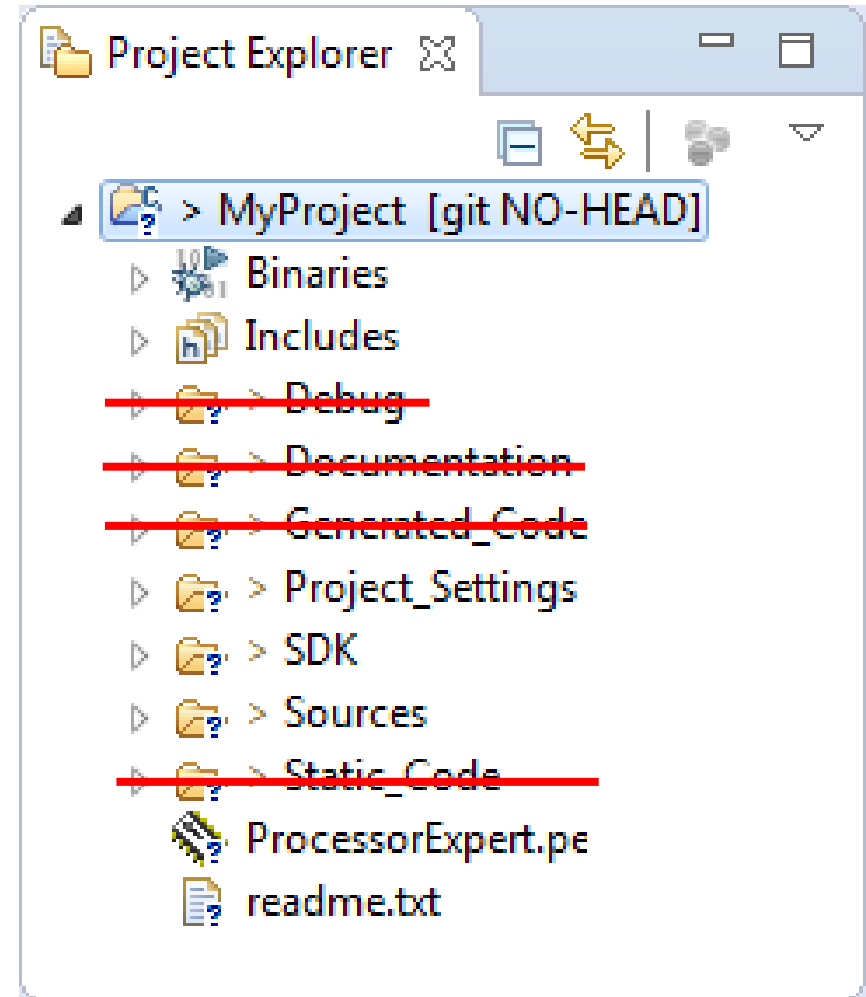
- Select repository previously created
- Press 'Finish'

- NOTE: Project will get **moved** to repository folder as working copy!



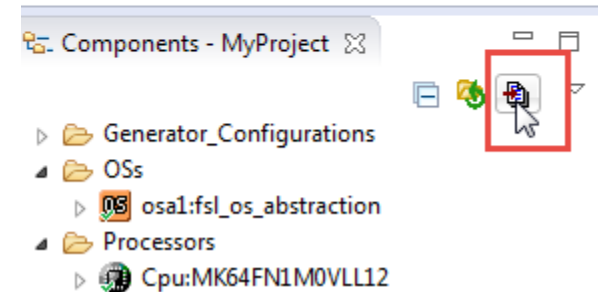
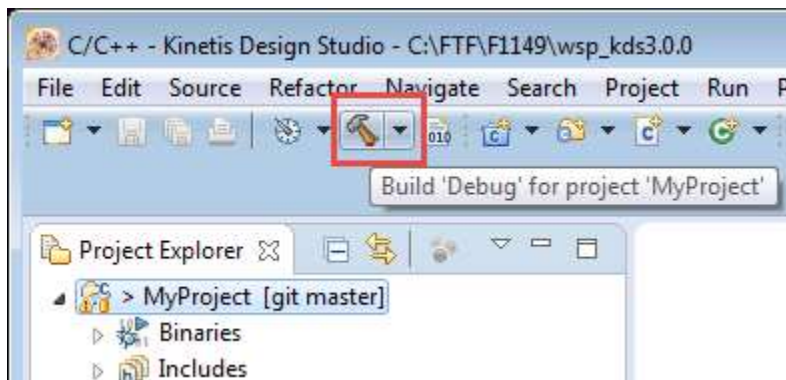
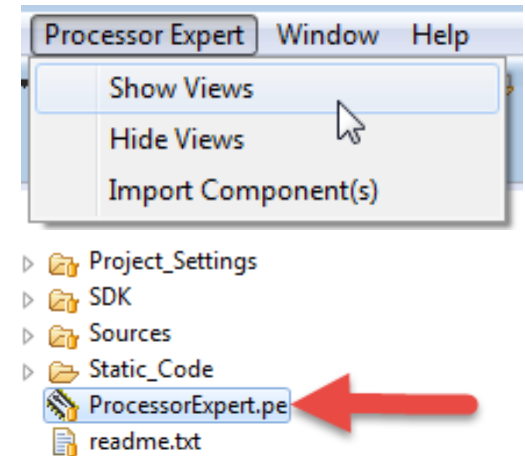
Project Explorer View

- Files/Folders marked in Project Explorer view
 - '?' means that item has unknown state/not in repository
- As project has moved, delete generated files/folders



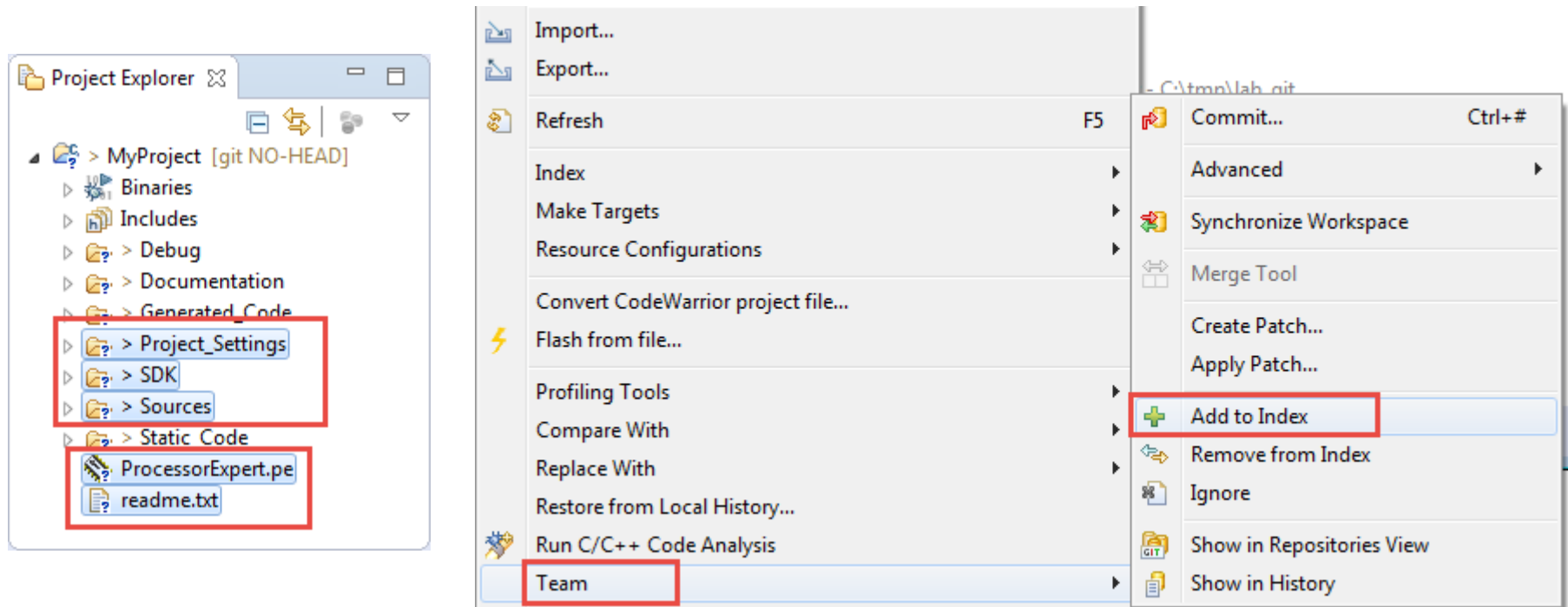
Generating Code, Build

- Menu *Processor Expert* > *Show Views*
- Select .pe File
- Use 'Generate Code' icon
- Select Project
- Build project



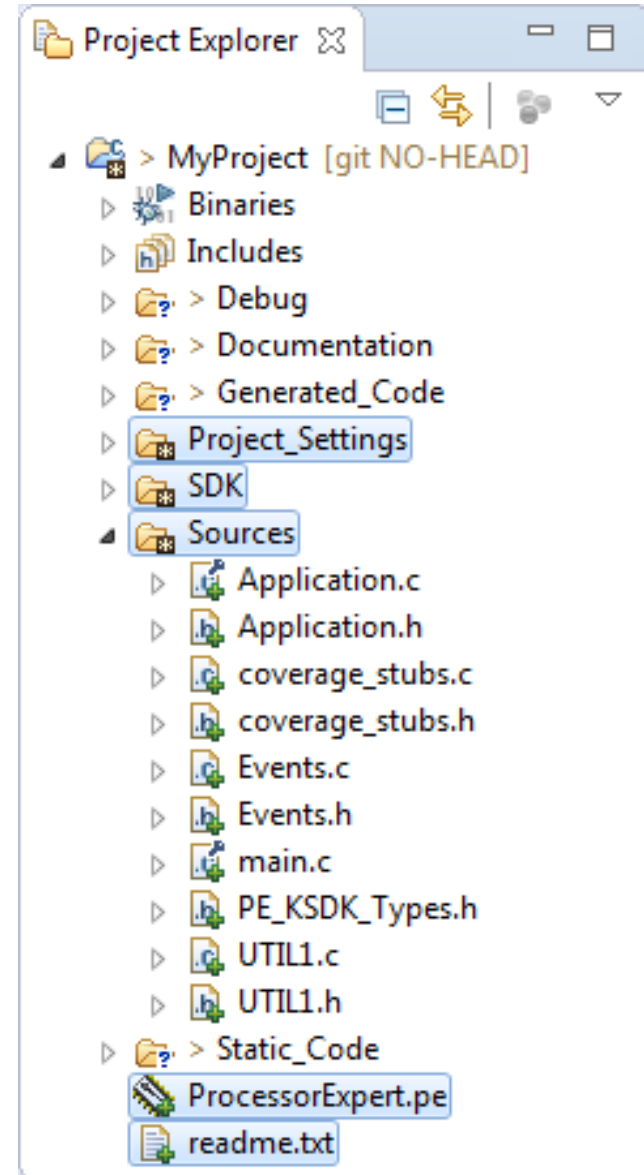
Adding Files to Index

- Select File/Folders to be added
- Context menu: *Team* > *Add to Index*



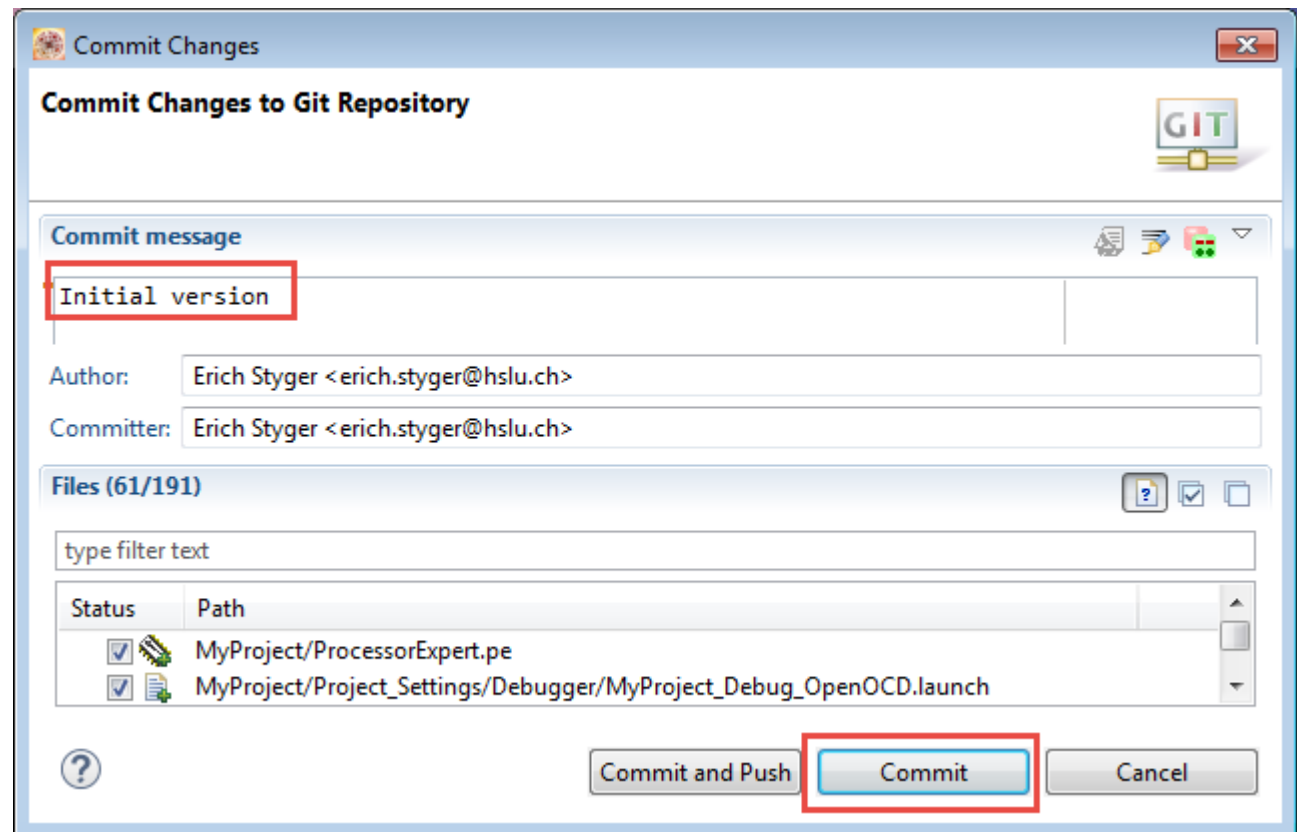
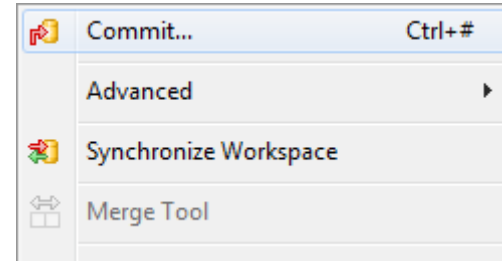
Added Files

- Newly added files get a '+' decorator
- Files/Folders changed have a '*' decorator
- The have been 'added', but are not 'committed' yet



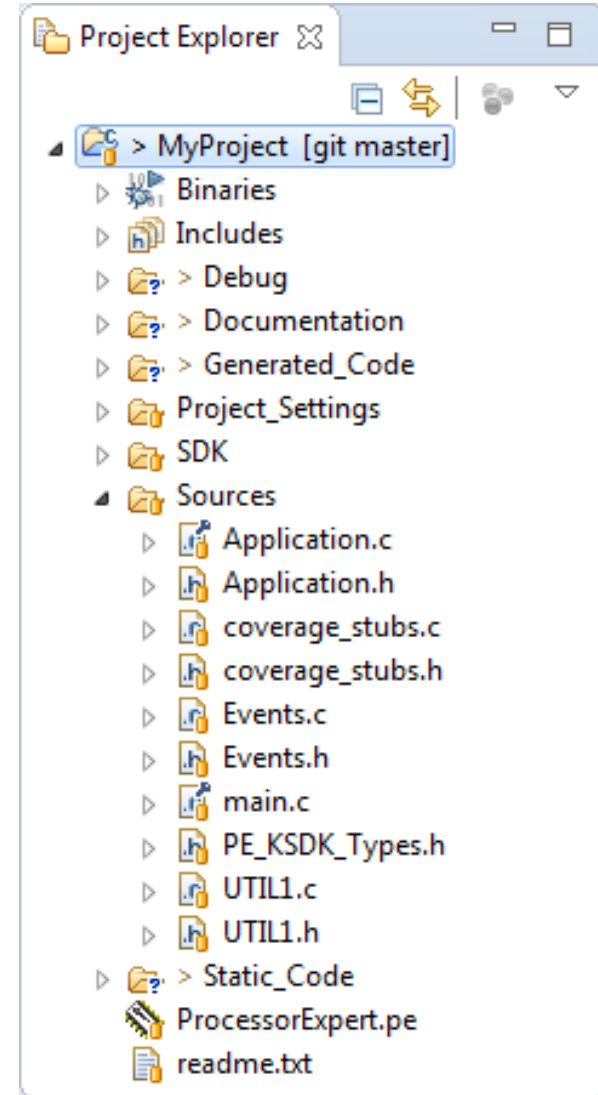
Commit Changes

- Use Context menu
Team > Commit
- Provide commit message
- **Commit** (to local repository)
- If using remote repository: Push



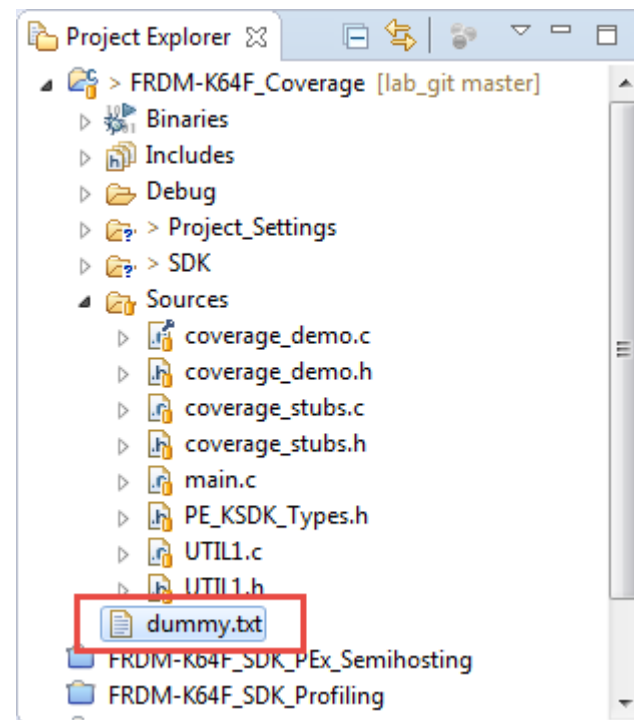
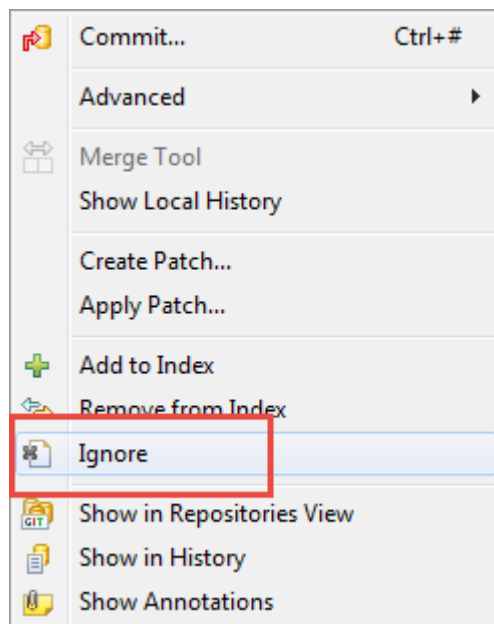
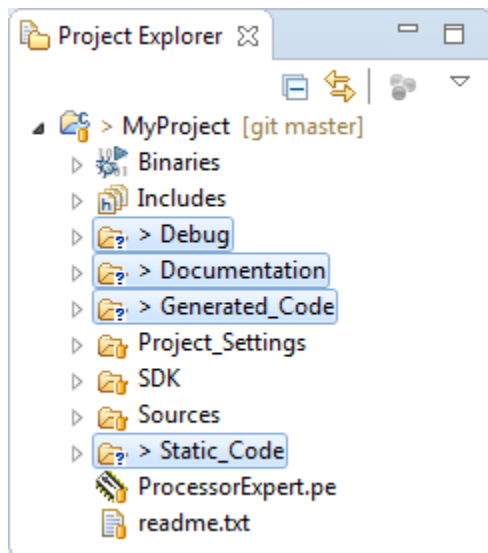
Committed Files

- Files in (local/remote) Repository get 'disk' decorator
- Files with '?' are still 'unknown' to the version control system
 - Need to 'ignore' them



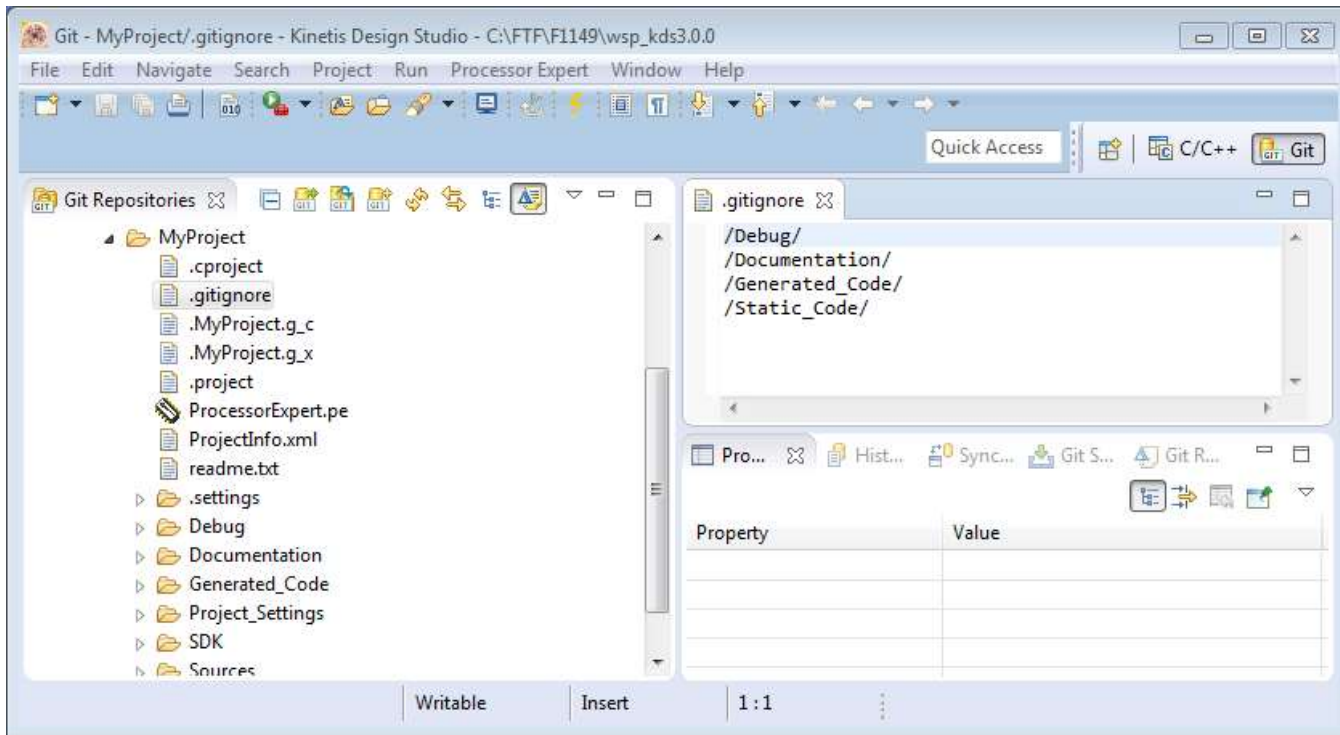
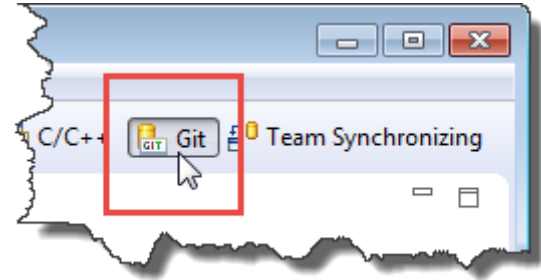
Ignoring Files and Folders

- Select files/folders to be ignored
- Context Menu *Team* > *Ignore*
- Will have '?' icon decorator removed



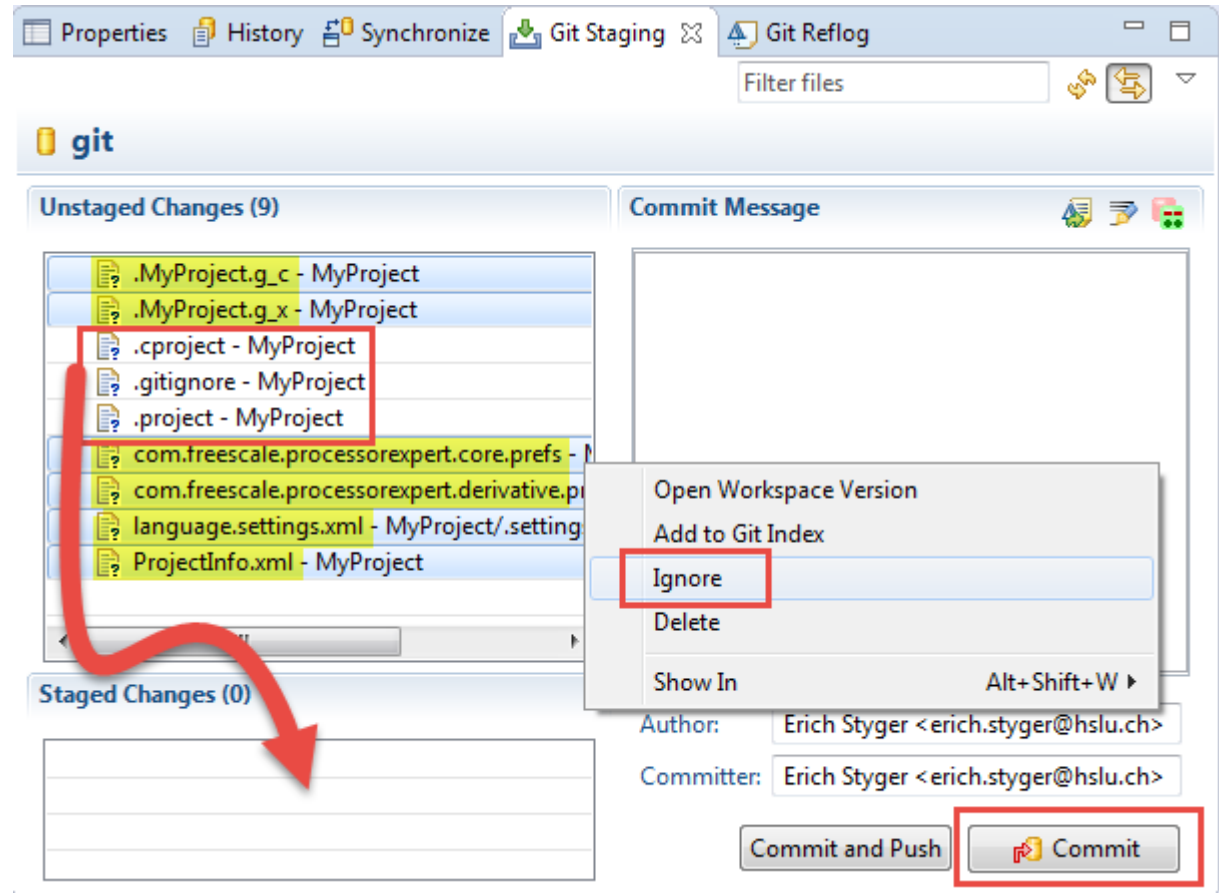
Commit: Ignored Files

- Switch to Git Perspective
- Ignored files/folders listed in .gitignore file
- Text file
- can have rules or edited with text editor



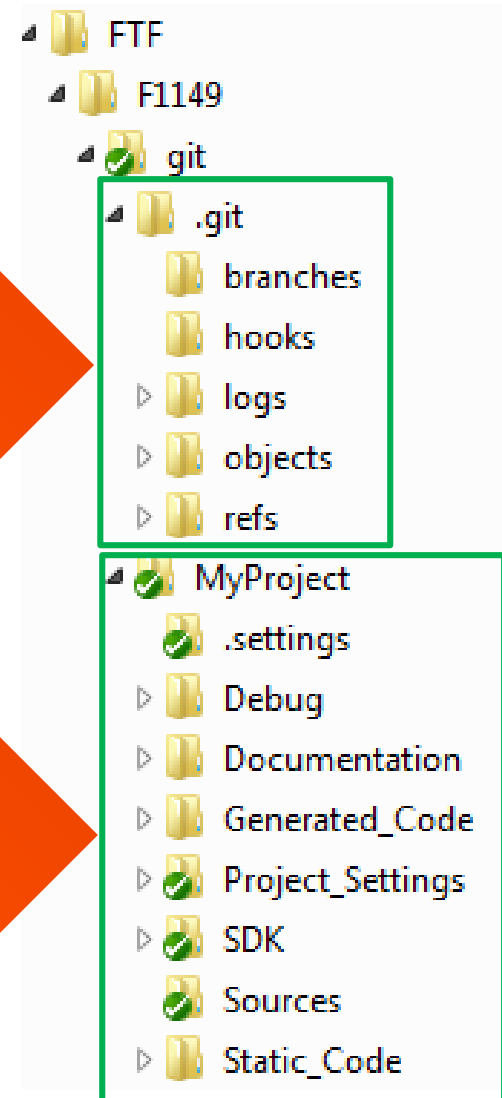
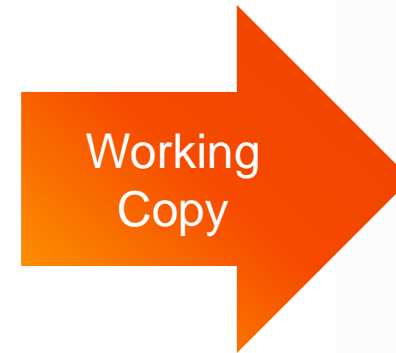
Git Staging

- Not all files are shown in Project view!
- Staging View in Git Perspective
- Drag&Drop files to Stage (add)
- Ignore File
- Double click to diff
- **Commit** with commit message



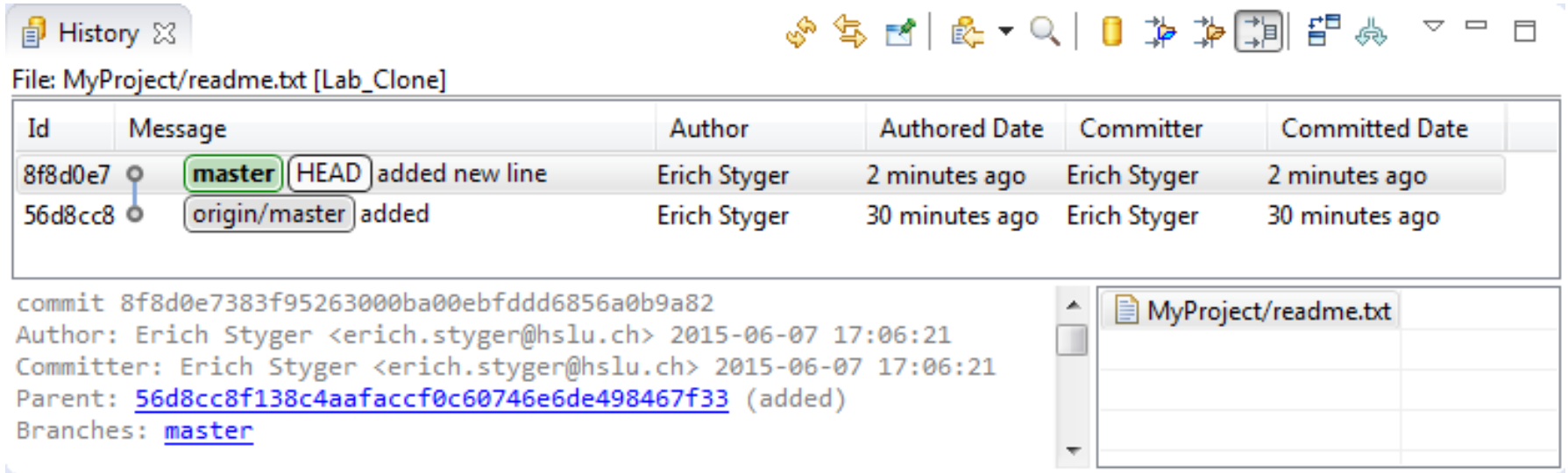
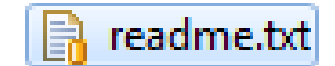
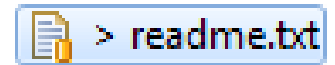
Local Git (not bare) Repository

- Git Repository/Database in .git Folder
 - Do not touch it unless you know what you are doing!
- With the Import into Git, our project has moved as 'working copy' into the local repository structure
- *NOTE: a **local** repository like this is not intended be shared with other developers. For this, a special 'bare' repository followed by a 'clone' would be necessary.*



History

- Edit a file and save it
 - Gets '>' marker to indicate change
- Commit it with context menu: *Team > Commit*
- Context menu *Team > Show in History*
 - History of files and changes
 - Double-click on line to open that version



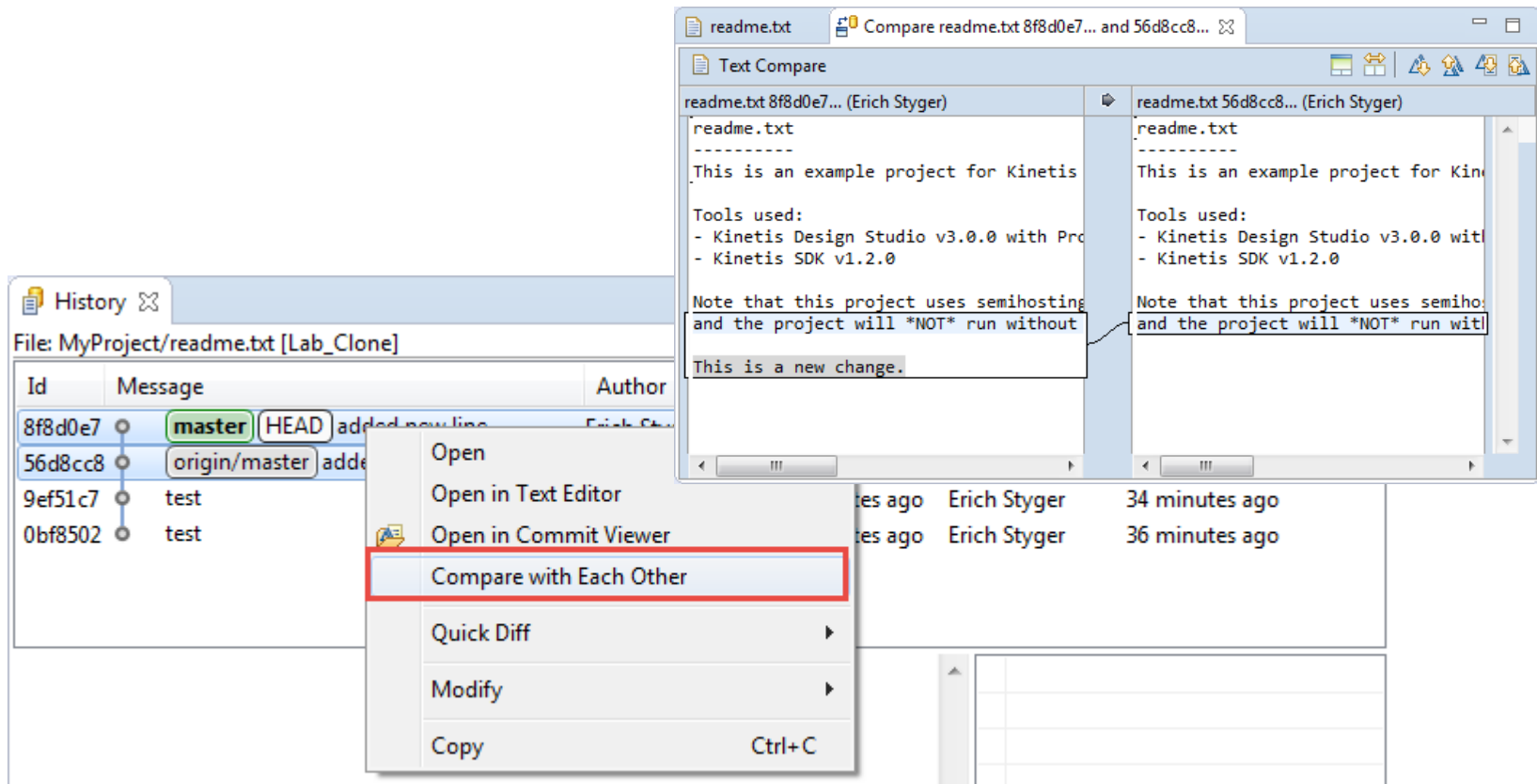
Id	Message	Author	Authored Date	Committer	Committed Date
8f8d0e7	master HEAD added new line	Erich Styger	2 minutes ago	Erich Styger	2 minutes ago
56d8cc8	origin/master added	Erich Styger	30 minutes ago	Erich Styger	30 minutes ago

```
commit 8f8d0e7383f95263000ba00ebfddd6856a0b9a82
Author: Erich Styger <erich.styger@hslu.ch> 2015-06-07 17:06:21
Committer: Erich Styger <erich.styger@hslu.ch> 2015-06-07 17:06:21
Parent: 56d8cc8f138c4aafaccf0c60746e6de498467f33 (added)
Branches: master
```



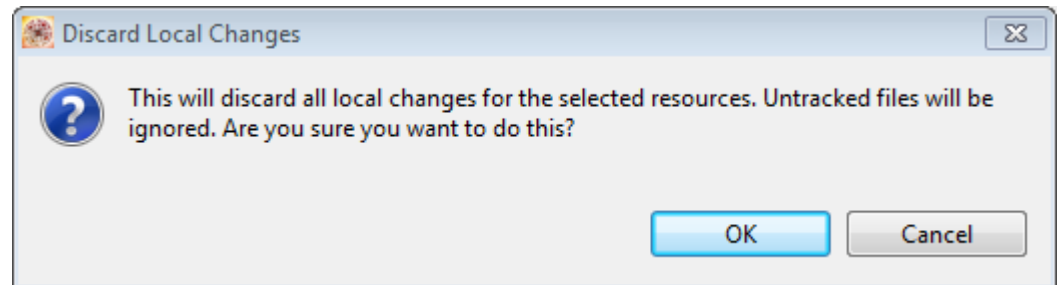
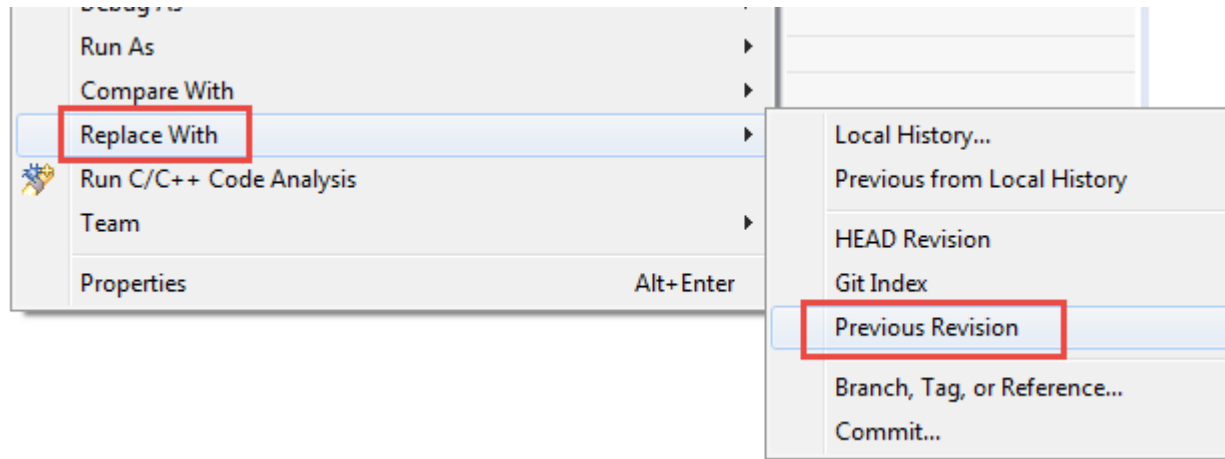
Compare

- Select two lines
- Context menu *Compare with Each Other*



Short Term Undo

- Select File
- Menu *Replace With* > *Previous Revision*



Summary

- Kinetis Design Studio IDE comes with version control built-in
- Understanding files in project
- Create Repository
- Add/Ignore/Change/Undo/Commit

- Further information:
 - <http://mcuoneclipse.com/2013/03/29/version-control-with-processor-expert-projects/>
 - <http://mcuoneclipse.com/2013/08/13/installing-egit-in-eclipse-and-codewarrior-for-mcu10-4/>
 - <http://mcuoneclipse.com/2012/03/17/dissection-of-mcu10-projects/>



Build Automation



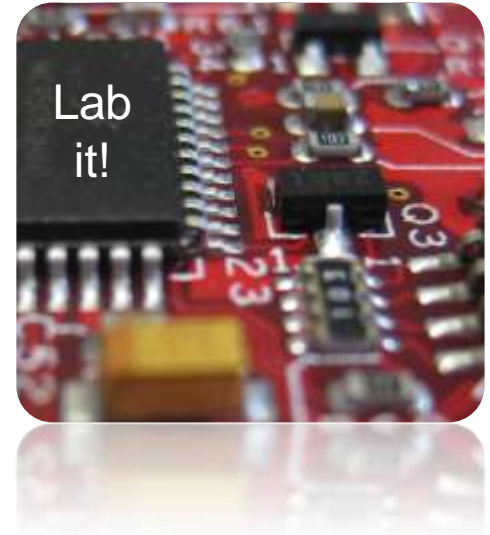
Why Build Automation

- Daily/Weekly/Monthly/etc automated builds
- In combination with version control system
 - Server checks out projects
 - Builds projects
 - Report errors/failures
- 'Continuous Integration' Process
- No graphical UI needed
 - Make files
 - Batch files
 - Scripting systems



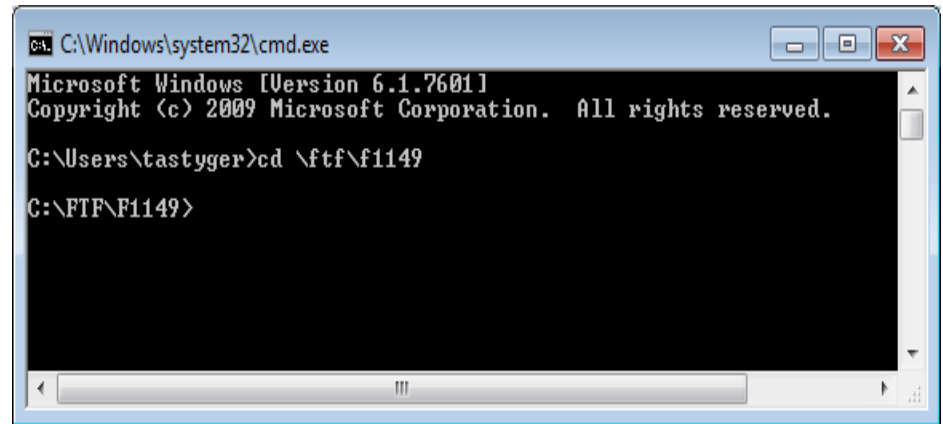
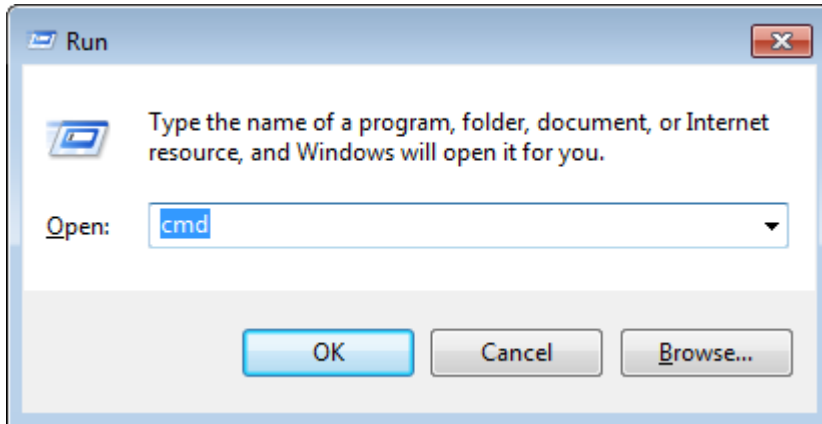
Build Automation: Lab Outline

- Build automation with Eclipse projects
- Using headless version of Eclipse
 - Generating Processor Expert Code
 - Generating make files
 - Building project
- Building project with make outside of Eclipse



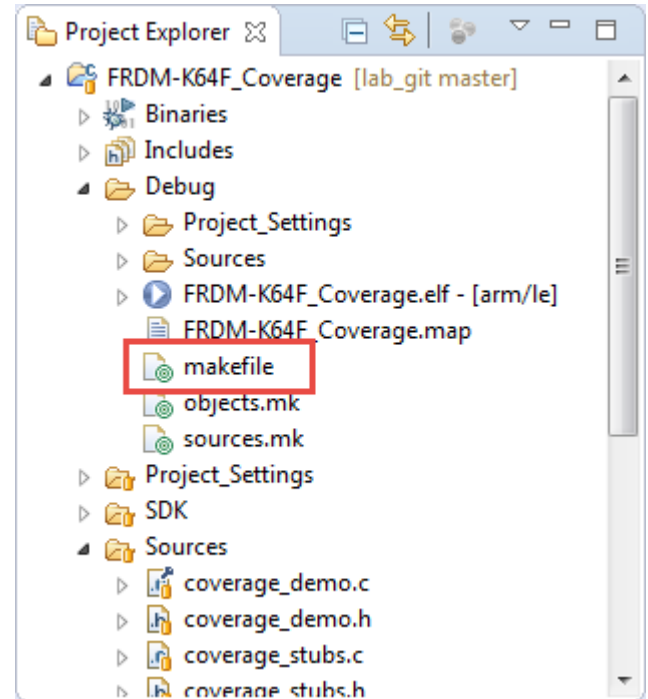
Windows Cmd.exe / DOS Shell

- Will use windows shell to run batch files in this lab
 - Simple
 - More sophisticated scripting: Python, TCL, Perl, Make Files, ...
 - Principles applicable for other hosts (Mac OS, Linux)
- <Windows-R> (Run), then 'cmd' to open Cmd.exe
- **cd \ftf\f1149**



PATH Setup for Labs

- Eclipse build system uses make files
 - 'manual' make files
 - 'managed' make files (default in KDS)
- KDS v3.0.0 has make here:
 - C:\Freescale\KDS_3.0.0\bin
- Need to have GNU tools in PATH
 - C:\Freescale\KDS_3.0.0\toolchain\bin



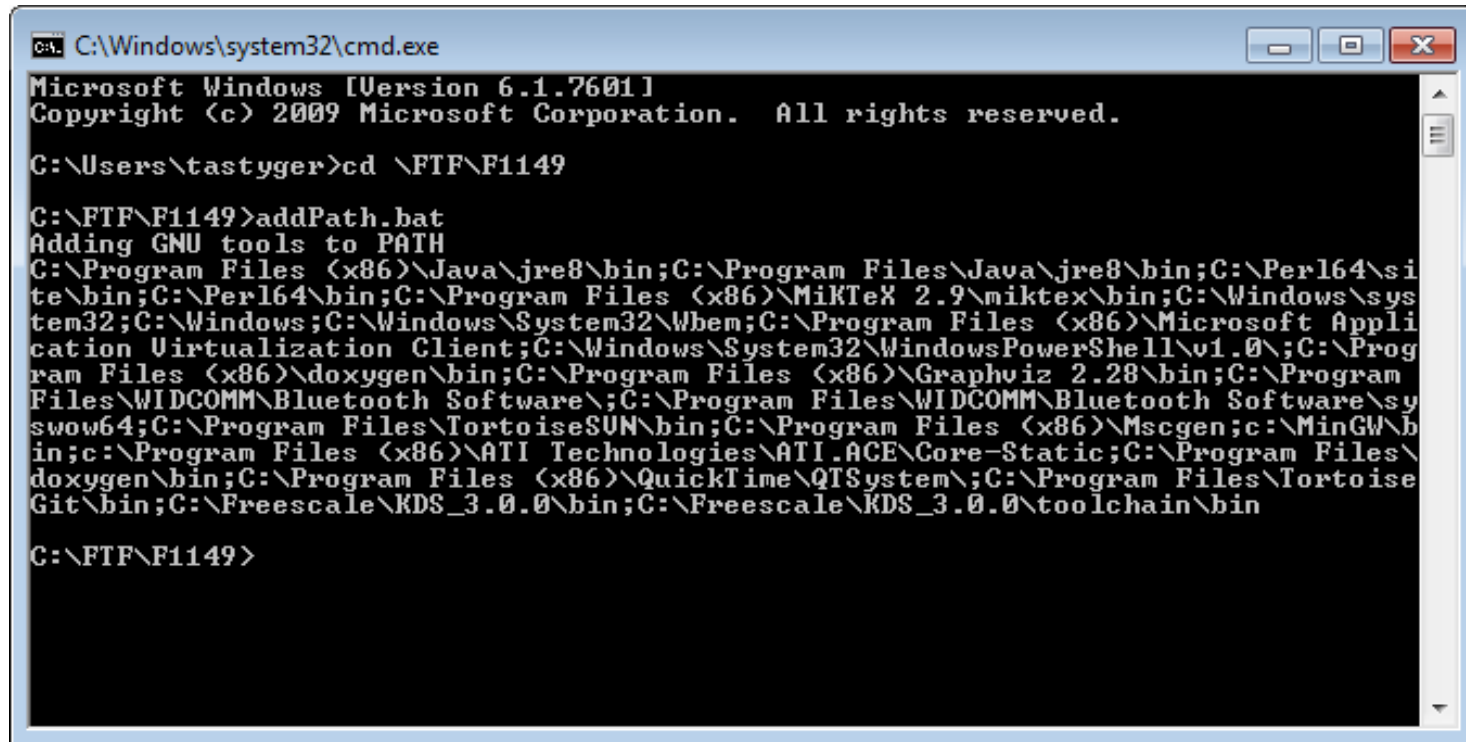
PATH Setup for Labs

- Run `addPath.bat`

- Extended PATH with

```
c:\Freescale\KDS_3.0.0\bin;
```

```
c:\Freescale\KDS_3.0.0\toolchain\bin
```



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

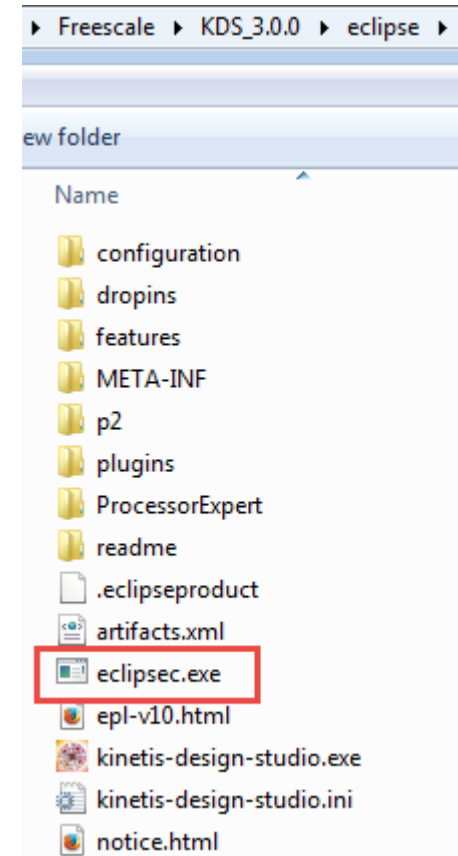
C:\Users\tastyger>cd \FTF\F1149

C:\FTF\F1149>addPath.bat
Adding GNU tools to PATH
C:\Program Files (x86)\Java\jre8\bin;C:\Program Files\Java\jre8\bin;C:\Perl64\site\bin;C:\Perl64\bin;C:\Program Files (x86)\MiKTeX 2.9\miktex\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Program Files (x86)\Microsoft Application Virtualization Client;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\doxygen\bin;C:\Program Files (x86)\Graphviz 2.28\bin;C:\Program Files\WIDCOMM\Bluetooth Software\;C:\Program Files\WIDCOMM\Bluetooth Software\swsow64;C:\Program Files\TortoiseSUN\bin;C:\Program Files (x86)\Mscgen;c:\MinGW\bin;c:\Program Files (x86)\ATI Technologies\ATI.ACE\Core-Static;C:\Program Files\doxygen\bin;C:\Program Files (x86)\QuickTime\QTSystem\;C:\Program Files\TortoiseGit\bin;C:\Freescale\KDS_3.0.0\bin;C:\Freescale\KDS_3.0.0\toolchain\bin

C:\FTF\F1149>
```

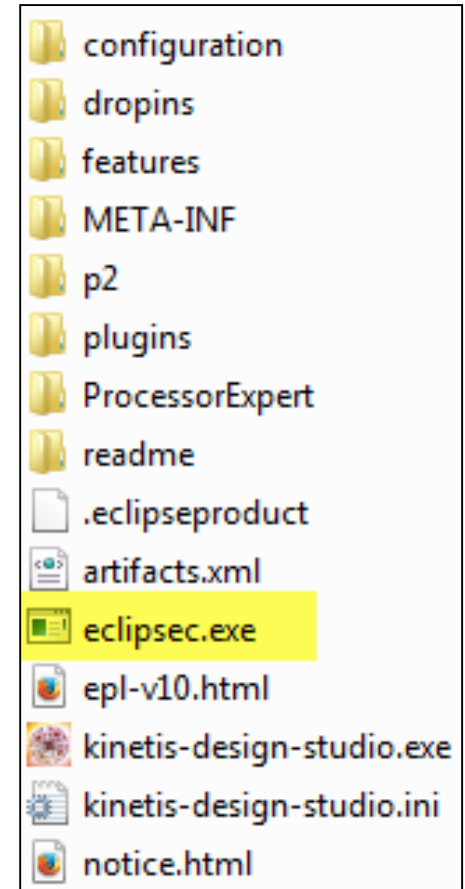

Building with Headless Eclipse

- Problem
 - Make files need to be generated
 - Generated files are usually not in version control system
- Solution
 - 'Headless' Build
 - Run Eclipse from the command line



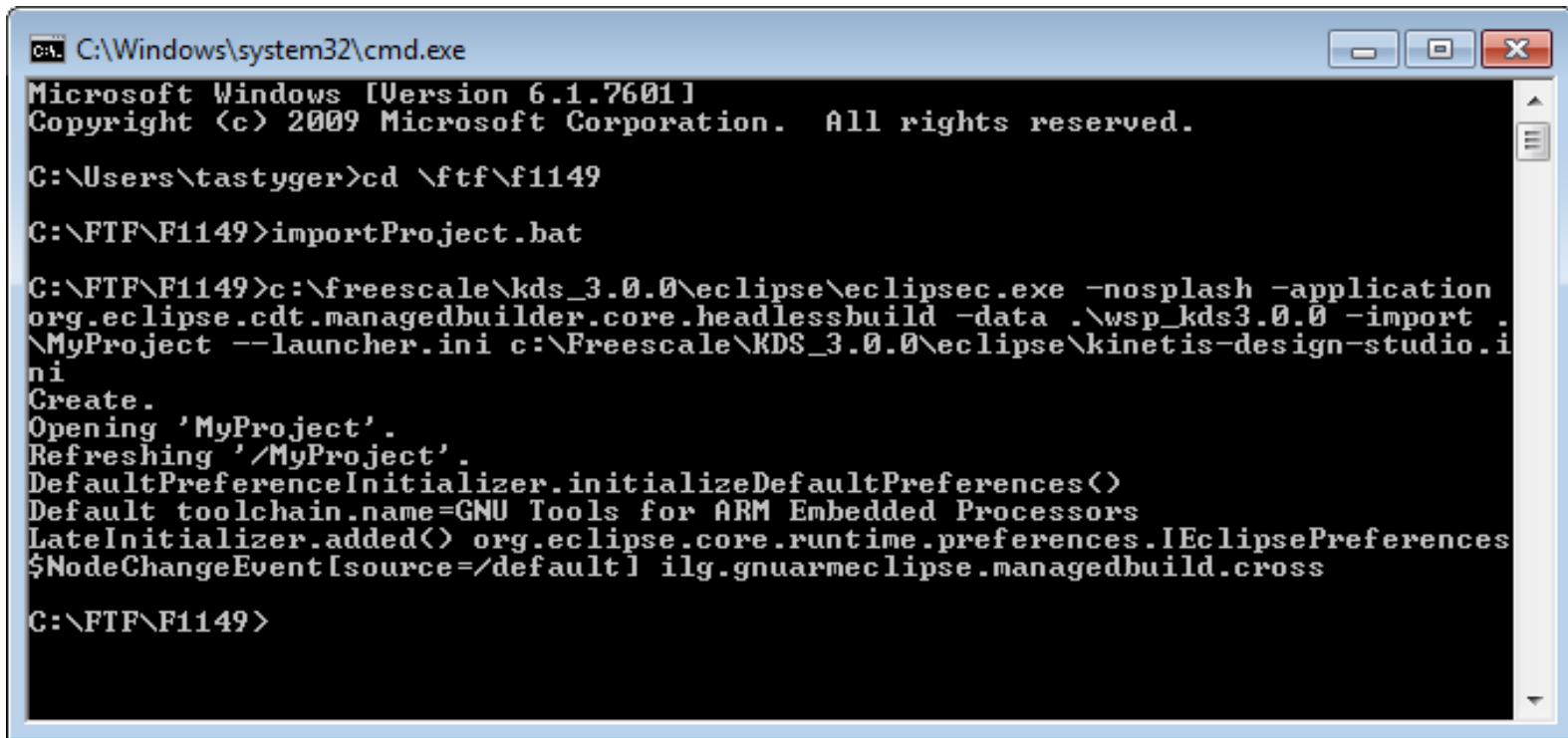
Eclipse Command Line Mode: Import Project

- **Eclipsec**: command line version of Eclipse
- Command Line Options
 - nosplash: avoid splash screen
 - launcher.suppressErrors: no error
 - application <plugin>: plugin to run
org.eclipse.cdt.managedbuilder.core.headlessbuild
com.freescale.processorexpert.core.PExApplication
 - import <project>: import project into workspace
 - data <wsp>: workspace to use
- Build:
 - build <project>: project to build
 - cleanbuild all: build all projects
 - launcher.ini <inifile>: JRE, heap settings
 - launcher.suppressErrors: no error dialogs
- Options can be combined/use separately



Importing Project

- Run `importProject.bat`
 - Creates workspace
 - Imports Project into workspace



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\tastyger>cd \ftf\f1149

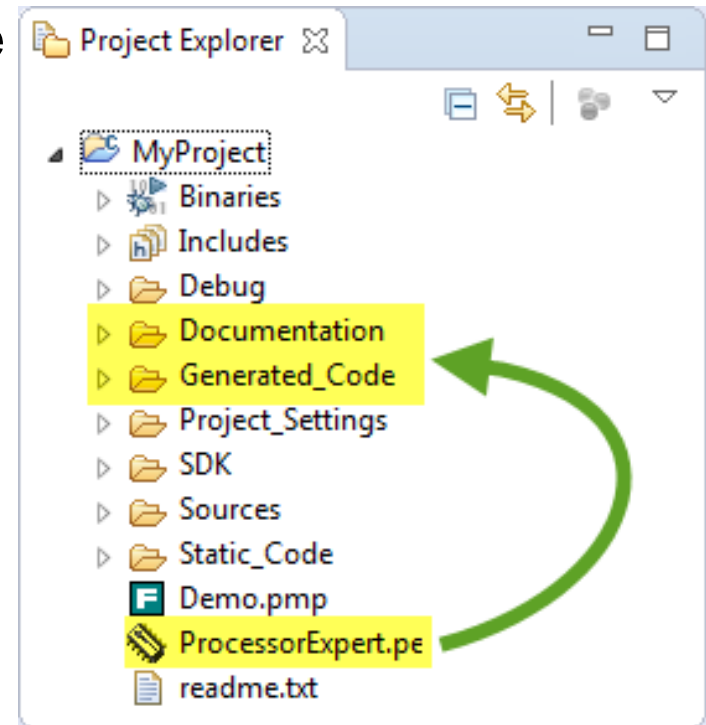
C:\FTF\F1149>importProject.bat

C:\FTF\F1149>c:\freescale\kds_3.0.0\eclipse\eclipsec.exe -nosplash -application
org.eclipse.cdt.managedbuilder.core.headlessbuild -data .\wsp_kds3.0.0 -import .
\MyProject --launcher.ini c:\Freescale\KDS_3.0.0\eclipse\kinetis-design-studio.i
ni
Create.
Opening 'MyProject'.
Refreshing '/MyProject'.
DefaultPreferenceInitializer.initializeDefaultPreferences()
Default toolchain.name=GNU Tools for ARM Embedded Processors
LateInitializer.added() org.eclipse.core.runtime.preferences.IEclipsePreferences
$NodeChangeEvent[source=/default] ilg.gnuarmeclipse.managedbuild.cross

C:\FTF\F1149>
```

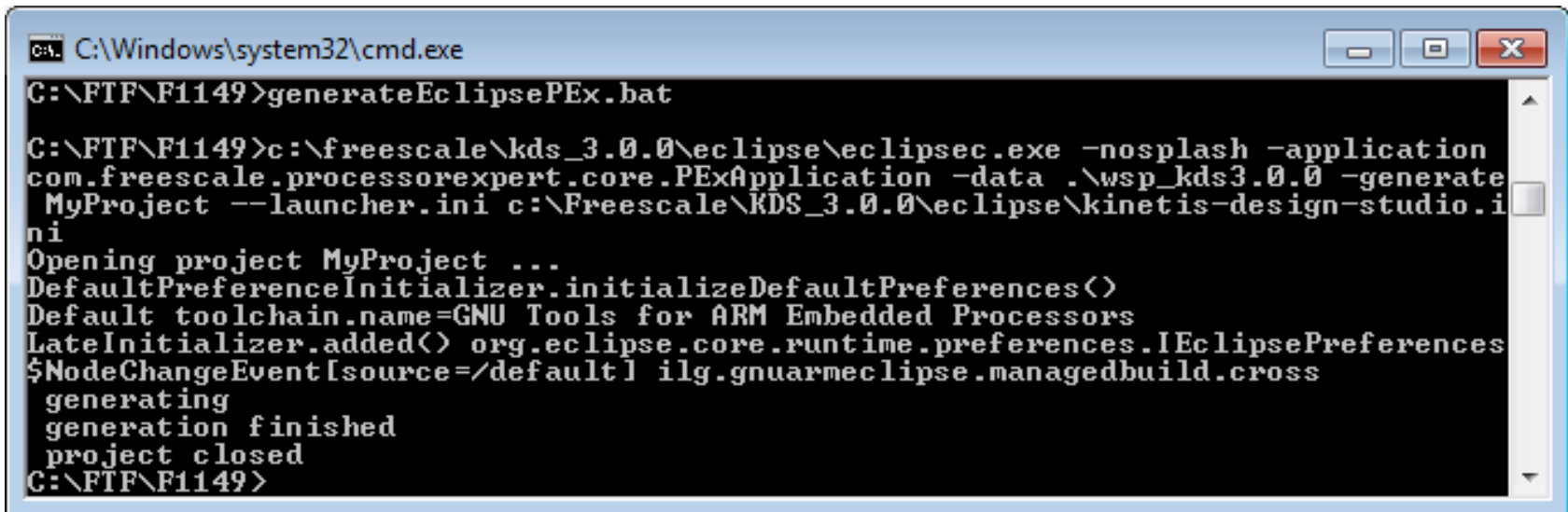
Processor Expert Projects and Eclipse

- Problem
 - Make files not present
 - If using Processor Expert: generates source code
 - Typically not stored in version control system
- Solution
 - Run Eclipse to generate Processor Expert code and to generate make files



Generating Processor Expert Code

- Only required if using Processor Expert
- Run **generateEclipsePEX.bat**
 - Generates Processor Expert code



```
C:\Windows\system32\cmd.exe
C:\FTP\F1149>generateEclipsePEX.bat
C:\FTP\F1149>c:\freescale\kds_3.0.0\eclipse\eclipsesec.exe -nosplash -application
com.freescale.processorexpert.core.PEXApplication -data .\wsp_kds3.0.0 -generate
MyProject --launcher.ini c:\Freescale\KDS_3.0.0\eclipse\kinetis-design-studio.i
ni
Opening project MyProject ...
DefaultPreferenceInitializer.initializeDefaultPreferences()
Default toolchain.name=GNU Tools for ARM Embedded Processors
LateInitializer.added() org.eclipse.core.runtime.preferences.IEclipsePreferences
$NodeChangeEvent[source=/default] ilg.gnuarmeclipse.managedbuild.cross
generating
generation finished
project closed
C:\FTP\F1149>
```

Build Project with Eclipse → Make

- Run `buildEclipse.bat`
 - Builds project

```
C:\Windows\system32\cmd.exe
rec.o ./Sources/freemaster_driver/freemaster_scope.o ./Sources/freemaster_driver
/freemaster_serial.o ./Sources/freemaster_driver/freemaster_sfio.o ./Sources/fre
emaster_driver/freemaster_tsa.o ./Sources/Events.o ./Sources/main.o ./SDK/plat
form/system/src/interrupt/fsl_interrupt_manager.o ./SDK/platform/system/src/clo
ck/MK64F12/fsl_clock_MK64F12.o ./SDK/platform/system/src/clock/fsl_clock_manage
r.o ./SDK/platform/system/src/clock/fsl_clock_manager_common.o ./SDK/platform/o
sa/src/fsl_os_abstraction_bm.o ./SDK/platform/hal/src/sim/MK64F12/fsl_sim_hal_M
K64F12.o ./SDK/platform/hal/src/rtc/fsl_rtc_hal.o ./SDK/platform/hal/src/port/
fsl_port_hal.o ./SDK/platform/hal/src/osc/fsl_osc_hal.o ./SDK/platform/hal/src
/mcg/fsl_mcg_hal.o ./SDK/platform/hal/src/mcg/fsl_mcg_hal_modes.o ./SDK/platfor
m/hal/src/gpio/fsl_gpio_hal.o ./SDK/platform/drivers/src/rtc/fsl_rtc_common.o
./SDK/platform/drivers/src/gpio/fsl_gpio_common.o ./SDK/platform/drivers/src/gpi
o/fsl_gpio_driver.o ./SDK/platform/devices/MK64F12/startup/gcc/startup_MK64F12.
o ./SDK/platform/devices/MK64F12/startup/system_MK64F12.o ./SDK/platform/devic
es/startup.o ./Generated_Code/Cpu.o ./Generated_Code/clockMan1.o ./Generated_Co
de/gpio1.o ./Generated_Code/hardware_init.o ./Generated_Code/osa1.o ./Generated
_Code/pin_init.o
Finished building target: MyProject.elf

Invoking: Cross ARM GNU Print Size
arm-none-eabi-size --format=berkeley "MyProject.elf"
  text  data  bss  dec  hex filename
 20616  120   3796  24532  5fd4 MyProject.elf
Finished building: MyProject.siz

08:33:59 Build Finished (took 8s.592ms)

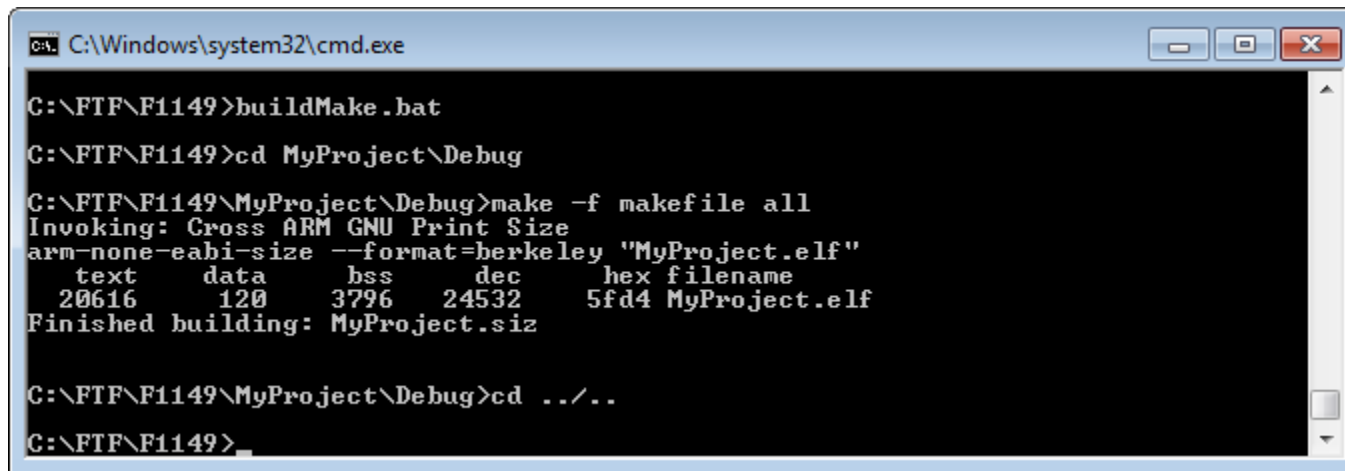
C:\FTF\F1149>
```



Build Project with Make

- Make `-f makefile all`
- Builds and links with managed make files (generated)
- Note: directory needs to be at make file level!
- Run `buildMake.bat`

```
cd MyProject\Debug
make -f makefile all
cd ../..
```



```
C:\Windows\system32\cmd.exe

C:\FTF\F1149>buildMake.bat
C:\FTF\F1149>cd MyProject\Debug
C:\FTF\F1149\MyProject\Debug>make -f makefile all
Invoking: Cross ARM GNU Print Size
arm-none-eabi-size --format=berkeley "MyProject.elf"
   text  data  bss   dec   hex filename
 20616   120   3796  24532  5fd4 MyProject.elf
Finished building: MyProject.siz

C:\FTF\F1149\MyProject\Debug>cd ../..
C:\FTF\F1149>
```

Overview

- Setup Environment
 - GNU Tools in PATH
 - addPath.bat
- Import Project into Workspace
 - Only if not already in workspace
 - importProject.bat
- Generate Processor Expert files
 - Only if Processor Expert Project
 - generateEclipsePEx.bat
- Build with Eclipse
 - Creates managed make files and builds
 - buildEclipse.bat
- Build with make (without Eclipse)
 - Only if make files exist
 - buildMake.bat

Setup Environment

Import Eclipse
Project

Generate Processor
Expert files

Create make files

Build with
Eclipse+Make

Build with Make

Summary

- Building Eclipse projects from the command line
- PATH to make/GNU tools
- Cmd.exe, Make.exe
- Make files need to be created/present
- Generating Processor Expert code
- Building with Eclipse
- Building with Make

- Further information:
 - <http://mcuoneclipse.com/2014/09/12/building-projects-with-eclipse-from-the-command-line/>
 - <http://mcuoneclipse.com/2013/10/26/eclipse-command-line-code-generation-with-processor-expert/>
 - <http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Freference%2Fmisc%2Fruntime-options.html>

Debug Automation



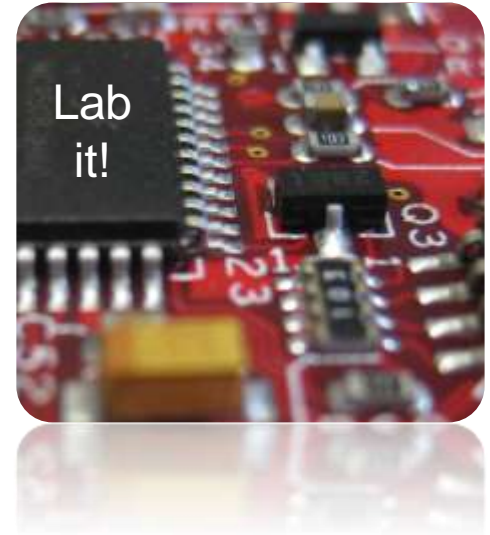
Why Debug Automation

- Daily/Weekly/Monthly/etc automated builds with debug
- In combination with version control system
 - Server checks out projects
 - Builds projects
 - Runs it on hardware
- 'Continuous Integration and Debug' Process
- Hardware/Test Farm
- No graphical UI needed
 - Make files
 - Batch files
 - Scripting systems



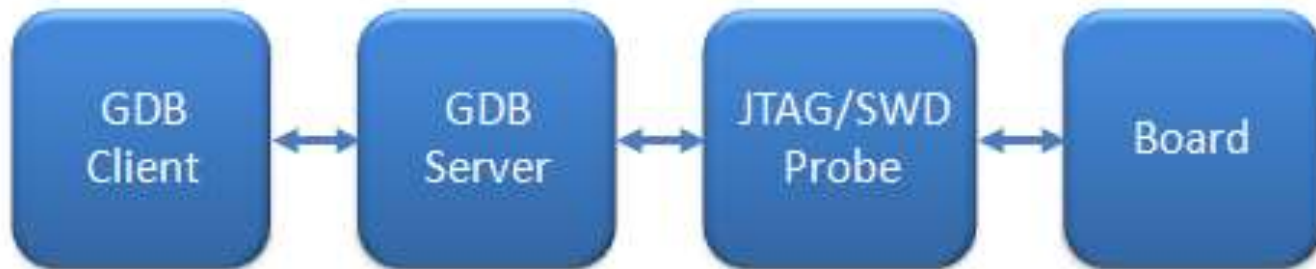
Debug Automation: Lab Outline

- Debug Eclipse projects without Eclipse
 - Command line GDB
- Exploring GDB
 - Running server/client
 - Load and debug application
- Automate debug session
 - Script file
 - Load and run application on board



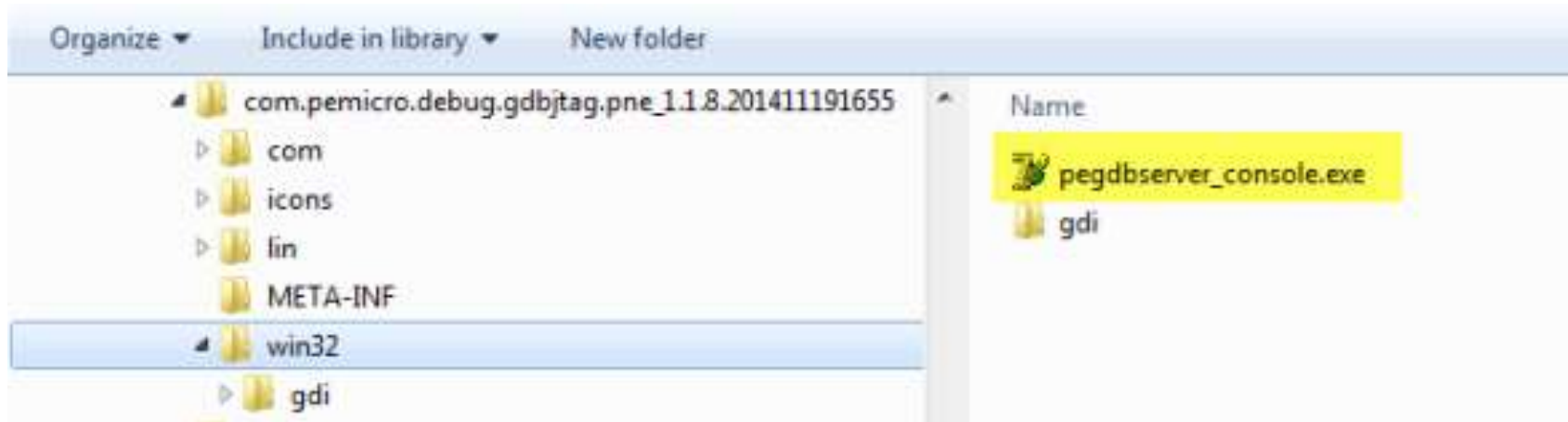
GDB Debugging

- GDB Client: communicates with server
 - arm-eabi-none-gdb.exe
- GDB Server: translates client commands into debug probe commands
 - P&E, Segger, OpenOCD, ...
- JTAG/SWD Probe
 - Segger J-Link, P&E Multilink, ...
- Board
 - FRDM-KL64F, FRDM-KL25Z, ...



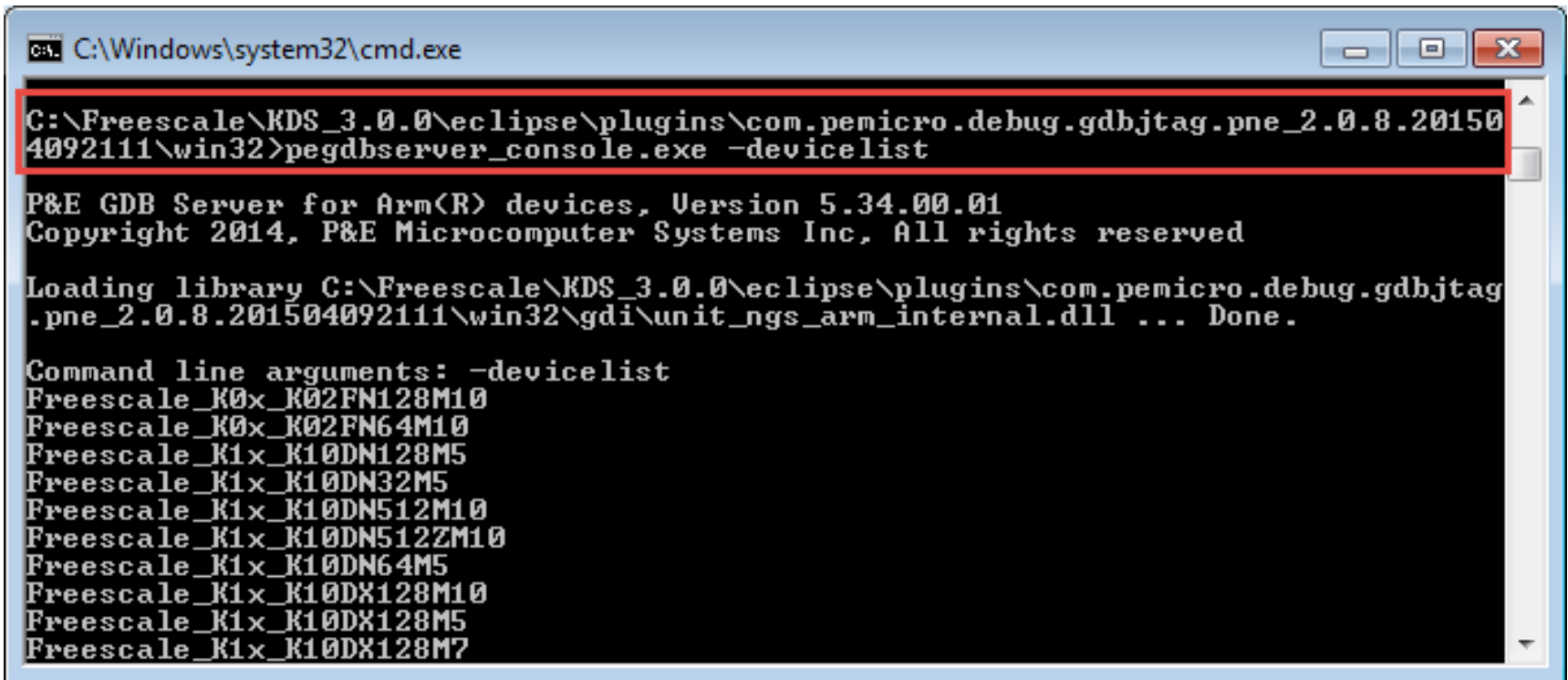
P&E GDB Server

- C:\Freescale\KDS_3.0.0\eclipse\plugins\com.pemicro.debug.gdbjtag.pne_2.0.8.201504092111\win32\pegdbserver_console.exe
- Command Line Arguments
 - **-startserver**: starts the server
 - **-device**: specifies the device
 - **-devicelist**: lists the supported devices



P&E GDB Server: Getting Device List

- `pegdbserver_console.exe -devicelist`



```
C:\Windows\system32\cmd.exe
C:\Freescall\KDS_3.0.0\eclipse\plugins\com.pemicro.debug.gdbjtag.pne_2.0.8.201504092111\win32>pegdbserver_console.exe -devicelist

P&E GDB Server for Arm(R) devices, Version 5.34.00.01
Copyright 2014, P&E Microcomputer Systems Inc, All rights reserved

Loading library C:\Freescall\KDS_3.0.0\eclipse\plugins\com.pemicro.debug.gdbjtag.pne_2.0.8.201504092111\win32\gdi\unit_ngs_arm_internal.dll ... Done.

Command line arguments: -devicelist
Freescale_K0x_K02FN128M10
Freescale_K0x_K02FN64M10
Freescale_K1x_K10DN128M5
Freescale_K1x_K10DN32M5
Freescale_K1x_K10DN512M10
Freescale_K1x_K10DN512ZM10
Freescale_K1x_K10DN64M5
Freescale_K1x_K10DX128M10
Freescale_K1x_K10DX128M5
Freescale_K1x_K10DX128M7
```

Start P&E GDB Server – Server stays running

- Have Board with P&E OpenSDA firmware plugged in
- Start new **cmd.exe**, cd to \FTF\F1149
- Run **gdbStartServerPnE.bat**
- Keep Server running!! It is using localhost:7224

```
ca: C:\Windows\system32\cmd.exe - StartPnEserver.bat
Loading library C:\Freescale\KDS_3.0.0\ eclipse\plugins\com.pemicro.debug.gdbjtag
.pne_2.0.8.201504092111\win32\gdi\unit_ngs_arm_internal.dll ... Done.
Command line arguments: -startserver -device=Freescale_R6x_K64FN1M0M12
Device selected is Freescale_R6x_K64FN1M0M12
HW Auto-Selected : Interface=OPENSDA Port=684A3E0C ; USB1 : OpenSDA (684A3E0C)

Connecting to target.
OpenSDA detected - Flash Version 1.08
Device is Freescale_R6x_K64FN1M0M12.
Mode is In-Circuit Debug.
'Kineticis' is a registered trademark of Freescale.
(C)copyright 2012, P&E Microcomputer Systems, Inc. (www.pemicro.com)
API version is 101
OpenSDA detected - Flash Version 1.08
Device is Freescale_R6x_K64FN1M0M12.
Mode is In-Circuit Debug.
'Kineticis' is a registered trademark of Freescale.
(C)copyright 2012, P&E Microcomputer Systems, Inc. (www.pemicro.com)
API version is 101
Server running on 127.0.0.1:7224
```

Server running on 127.0.0.1:7224



Starting GDB Client

- Run **arm-none-eabi-gdb**
- GDB prompt: (gdb)
- Client and Server not connected (yet)

GDB Client:

```
C:\Windows\system32\cmd.exe - arm-none-eabi-gdb
C:\FTF\F1149>cd MyProject\Debug
C:\FTF\F1149\MyProject\Debug>make -f makefile all
Invoking: Cross ARM GNU Print Size
arm-none-eabi-size --format=berkeley "MyProject.elf"
  text  data  bss  dec  hex filename
20616  120   3796  24532  5fd4 MyProject.elf
Finished building: MyProject.siz

C:\FTF\F1149\MyProject\Debug>cd ../../
C:\FTF\F1149>arm-none-eabi-gdb
GNU gdb (GNU Tools for ARM Embedded Processors) 7.6.0.
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu
This is free software: you are free to change and redi
There is NO WARRANTY, to the extent permitted by law.
and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb)
```

GDB Server (P&E):

```
C:\Windows\system32\cmd.exe - gdbStartServerPnE.bat
Connecting to target.
OpenSDA detected - Flash Version 1.08
Device is Freescale_K6x_K64FN1M0M12.
Mode is In-Circuit Debug.
'Kineticis' is a registered trademark of Freescale.
(C)opyright 2012, P&E Microcomputer Systems, Inc.
API version is 101
OpenSDA detected - Flash Version 1.08
Device is Freescale_K6x_K64FN1M0M12.
Mode is In-Circuit Debug.
'Kineticis' is a registered trademark of Freescale.
(C)opyright 2012, P&E Microcomputer Systems, Inc.
API version is 101
```



GDB Client: Connect and Reset

- Connect to P&E GDB server

- (gdb) **target remote localhost:7224**

```
(gdb) target remote localhost:7224
Remote debugging using localhost:7224
0x000004d8 in ?? (<)
```

Server running on 127.0.0.1:7224
Connection from "127.0.0.1" via 127.0.0.1

- Reset target

- (gdb) **monitor reset**

```
(gdb) monitor reset
Command Executed successfully: reset
```

Server running on 127.0.0.1:7224
Connection from "127.0.0.1" via 127.0.0.1
CPU reset by debugger.

GDB Client: Load

- Load Application File

- (gdb) **load <elf file name>**

- NOTE: use forward (/) slashes for path!

```
(gdb) load MyProject/Debug/MyProject.elf
Loading section .interrupts, size 0x400 lma 0x0
Loading section .flash_config, size 0x10 lma 0x400
Loading section .text, size 0x4c70 lma 0x410
Loading section .ARM, size 0x8 lma 0x5080
Loading section .init_array, size 0x4 lma 0x5088
Loading section .fini_array, size 0x4 lma 0x508c
Loading section .data, size 0x70 lma 0x5090
Start address 0x4d8, load size 20736
Transfer rate: 5 KB/sec, 864 bytes/write.
(gdb)
```

```
CMD>RE
```

```
Initializing.
Target has been RESET and is active.
```

- Load Debug Symbols

- (gdb) **file <elf file name>**

- Use 'y' to load file if already debugged

```
(gdb) file MyProject/Debug/MyProject.elf
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
Reading symbols from C:\FTF\F1149\MyProject\Debug\MyProject.elf...done.
(gdb)
```

GDB Client: Run

- Stepping

- (gdb) **step**

```
(gdb) step
328          bl SystemInit
(gdb) _
```

- Run application

- (gdb) **continue**

```
(gdb) continue
Continuing.
```

- Stop application

- **CTRL+C** will stop it

```
Program received signal SIGINT, Interrupt.
0x0000038d2 in OSA_TimeGetMsec () at ../Generated_Code/osa1.c:143
143      }
(gdb) _
```

- Quit/terminate GDB

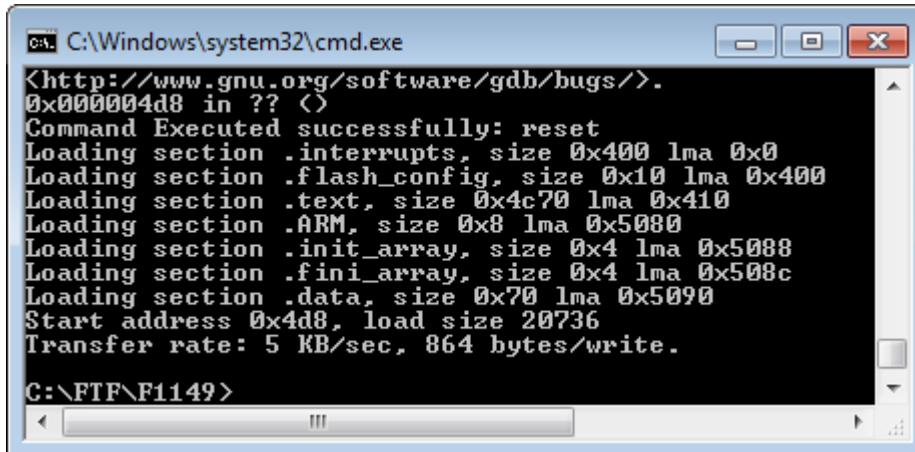
- (gdb) **quit**

```
(gdb) quit
A debugging session is active.

        Inferior 1 [Remote target] will be killed.
Quit anyway? (y or n) y
C:\FTF\F1149>
```

GDB Debug Automation

- Run everything from single batch file
 - Start GDB server (option `-single-session`)
 - Run gdb with script file (option `-x`)
- Script File
 - Set `confirm off`: disable confirmation question
 - Detach: detach from target and run it
- Close/Terminate running Server
- Run `gdbRun.bat`



```
C:\Windows\system32\cmd.exe
<http://www.gnu.org/software/gdb/bugs/>.
0x000004d8 in ?? (<)
Command Executed successfully: reset
Loading section .interrupts, size 0x400 lma 0x0
Loading section .flash_config, size 0x10 lma 0x400
Loading section .text, size 0x4c70 lma 0x410
Loading section .ARM, size 0x8 lma 0x5080
Loading section .init_array, size 0x4 lma 0x5088
Loading section .fini_array, size 0x4 lma 0x508c
Loading section .data, size 0x70 lma 0x5090
Start address 0x4d8, load size 20736
Transfer rate: 5 KB/sec, 864 bytes/write.
C:\FTF\F1149>
```

```
set confirm off
target remote localhost:7224
monitor reset
file MyProject/Debug/MyProject.elf
load MyProject/Debug/MyProject.elf
detach
quit
```

Summary

- GDB Client and GDB Server
- Command line debugging with GDB
- Scripting GDB debug session

- Further information:
 - <http://mcuoneclipse.com/2015/03/25/command-line-programming-and-debugging-with-gdb/>
 - <https://cs.brown.edu/courses/cs033/docs/guides/gdb.pdf>

Coverage



What is Coverage?

- Information about
 - Which parts have been executed in the code
 - How many times
- Test Coverage



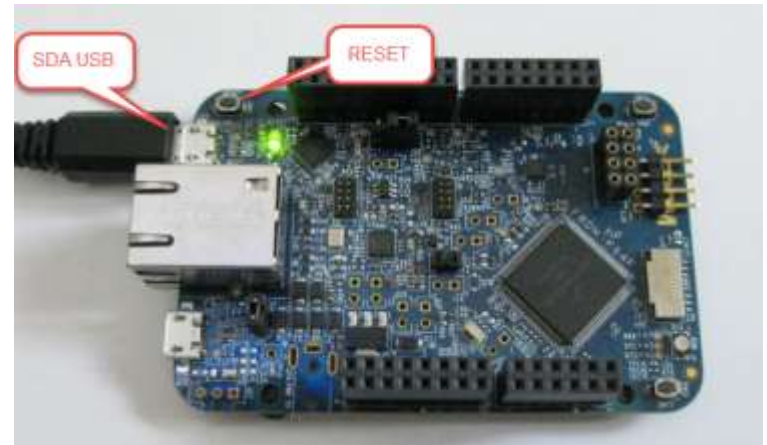
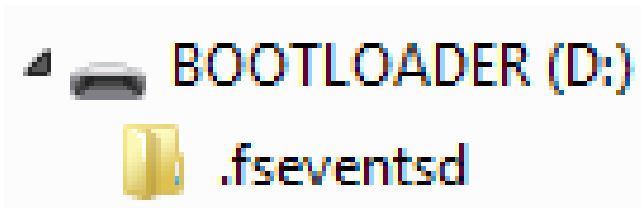
The screenshot shows a code editor window with the file 'Test.c' open. The code is as follows:

```
7  
8 #include "Test.h"  
9  
10 static int i;  
11  
12 static int bar(int i) {  
13     if (i==0) {  
14         return 2;  
15     } else if (i==1) {  
16         return 0;  
17     } else {  
18         return 1;  
19     }  
20 }  
21  
22 void TEST_Test(int val) {  
23     if (val==10) {  
24         i = val;  
25     } else {  
26         i += bar(i);  
27     }  
28 }  
29
```

The code is annotated with coverage bars on the left side of each line. The bars are colored green or red. The 'bar' function and the 'TEST_Test' function are mostly green, indicating they were executed. The 'return 2;' and 'return 0;' lines in the 'bar' function are red, indicating they were not executed. The 'return 1;' line in the 'bar' function is green, indicating it was executed. The 'if (val==10)' block in 'TEST_Test' is green, indicating it was executed. The 'else' block in 'TEST_Test' is red, indicating it was not executed.

Lab Setup: Segger FRDM-K64F OpenSDA Firmware

- Power Board with 'SDA USB' while RESET button pressed
 - Green LED near connector blinks slowly
 - Board enumerates as BOOTLOADER
 - Copy firmware file (**JLink_OpenSDA_V2_2015-04-23.bin**) to BOOTLOADER drive
 - Green LED blinks fast
- Repower board with SDA USB port normally
 - USB driver might enumerate
 - Green LED is on



Lab Setup: Gcov Binaries Dependency

- GNU binaries in C:\Freescale\KDS_3.0.0\toolchain\bin
- Have copies of needed GNU binaries without 'arm-none-eabi'
- Run **checkGNUbin.bat**
 - Copy of files with new names (without arm-none-eabi)

Name	Date modified	Type	Size
addr2line.exe	04.08.2014 22:52	Application	634 KB
arm-none-eabi-addr2line.exe	04.08.2014 22:52	Application	634 KB
arm-none-eabi-ar.exe	04.08.2014 22:52	Application	657 KB
arm-none-eabi-as.exe	04.08.2014 22:52	Application	657 KB
arm-none-eabi-gcc-nm.exe	04.08.2014 22:52	Application	50 KB
arm-none-eabi-gcc-ranlib.exe	04.08.2014 22:52	Application	50 KB
arm-none-eabi-gcov.exe	04.08.2014 22:52	Application	1'148 KB
arm-none-eabi-gdb.exe	04.08.2014 22:52	Application	4'407 KB
arm-none-eabi-gprof.exe	04.08.2014 22:52	Application	692 KB
arm-none-eabi-strip.exe	04.08.2014 22:52	Application	1'057 KB
gccvar.bat	04.08.2014 22:52	Windows Batch File	1 KB
gcov.exe	04.08.2014 22:52	Application	1'148 KB

Lab Setup: Launch Eclipse

- Command Prompt (<Windows>+R, then cmd)
- Change director to c:\ftf\1149
- Run **runEclipseGcov.bat**
 - Ensures GNU tools in PATH
 - Ensures proper names of GNU tools
 - Launches Kinetis Design Studio with workspace specified

```
REM Run Eclipse with gcov environment

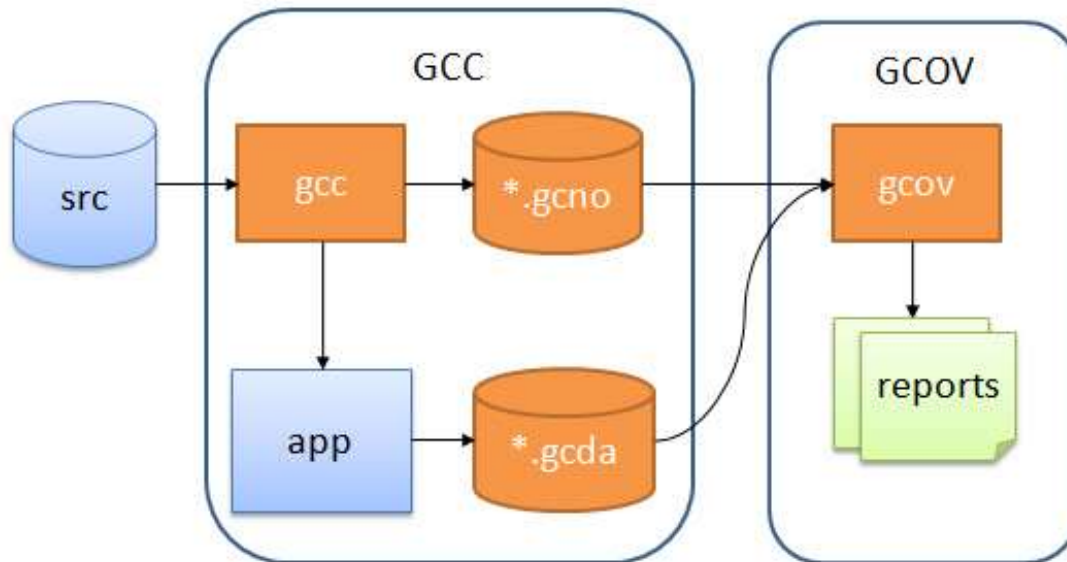
REM Check that path setup is correct
call addPath.bat

REM Check that GNU tools are present
call checkGNUbin.bat

REM launch Eclipse
C:\Freescale\KDS_3.0.0\eclipse\kinetis-design-studio.exe -data .\wsp_kds3.0.0
```

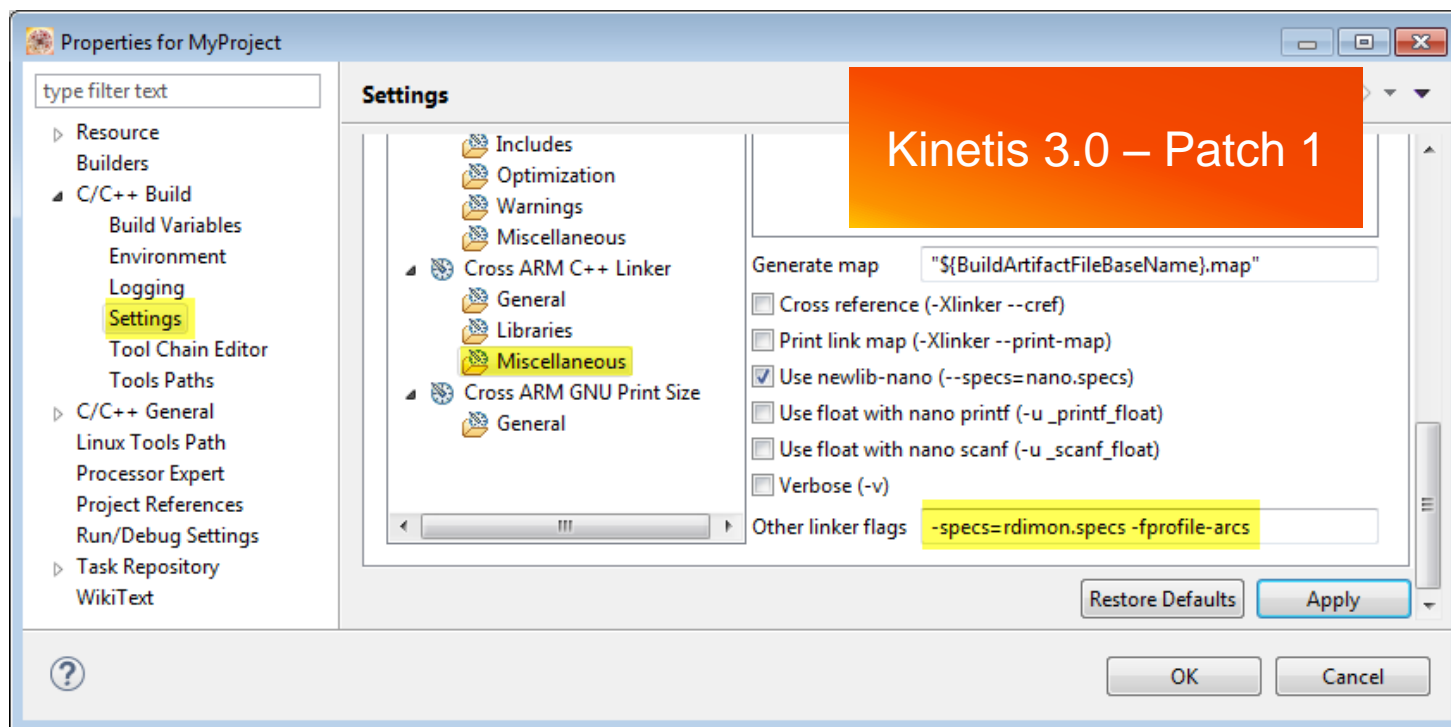
Overview: Coverage with GNU Tools

- Instrumentation of application with gcc
 - Generates .gcno (GNU Coverage Node) information
- Run the application on target
 - Generates .gcda (Gnu Coverage Data) file
- Show coverage information on host with gcov



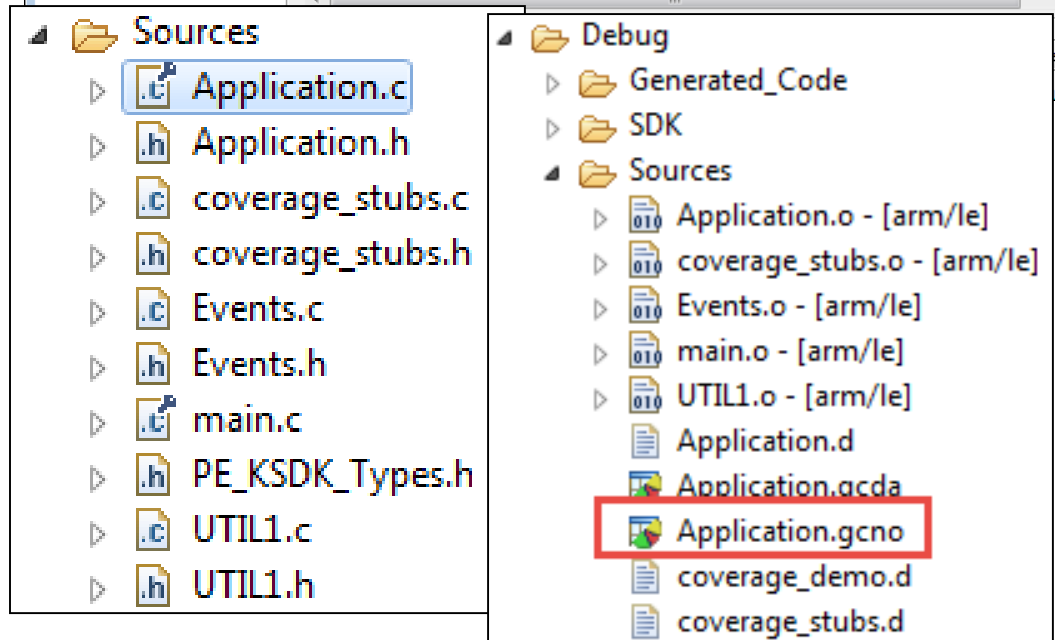
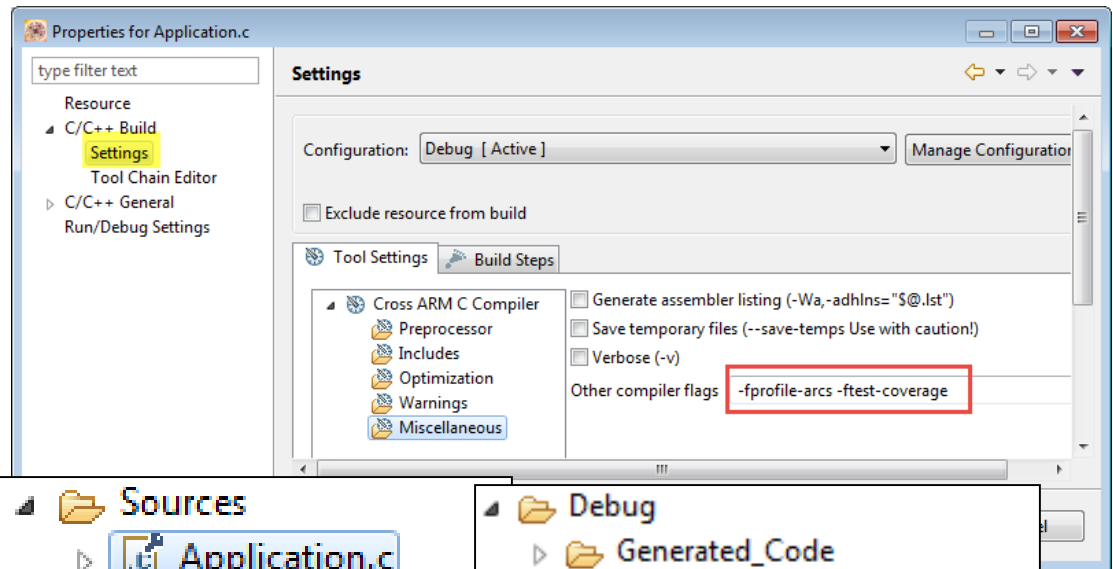
Instrumentation: Linker Project Option

- Project Properties
- Miscellaneous Linker Settings
 - specs=rdimon.specs: Use semihosting (file/console I/O)
 - fprofile-arcs: Use profiling/instrumentation



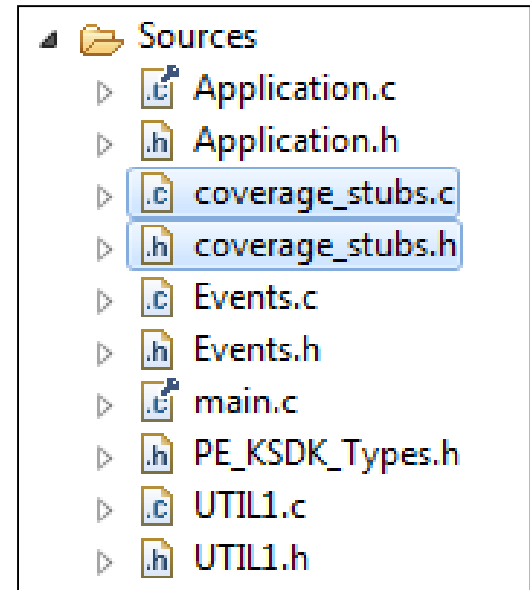
Instrumentation Compiler Option

- Instrument few files, less overhead/RAM needed
- Right click on file to instrument
- 'Pinned' mode
- Compiler options
 - fprofile-arcs
 - ftest-coverage
- Will create *.gcno file



Coverage Stubs

- Set of coverage support routines
 - `_exit()`, `_write()`, `_open()`, `_close()`, ...
- Coverage will write 'file' with data
 - Can intercept `_write()` to dump data with gdb to file (See Backup material)
 - Using **semihosting** to write data
 - `COV_USE_SEMIHOSTING` macro enabled in `coverage_stubs.h`



Coverage Information Written at Exit

- Application to call `_exit()`
- Coverage data appended to existing data files



```
Application.c
void APP_Run(void) {
    printf("Lab program using semihosting!\r\nProgram will NOT
    for(;;) {
        CheckButton();
        if (whichLED==LED_COLOR_RED) {
            GPIO_DRV_TogglePinOutput(led_red); /* red toggle */
        }
        if (whichLED==LED_COLOR_GREEN) {
            GPIO_DRV_TogglePinOutput(led_green); /* green toggle */
        }
        if (whichLED==LED_COLOR_BLUE) {
            GPIO_DRV_TogglePinOutput(led_blue); /* blue toggle */
        }
        OSA_TimeDelay(100); /* wait 100 ms */
        doExit = nofButtonPresses>5;
        if (doExit) {
            _exit(0); /* write coverage information */
        }
    }
}
```

```
coverage_stubs.c
void _exit(int status) {
    void gcov_exit(void); /* prototype */

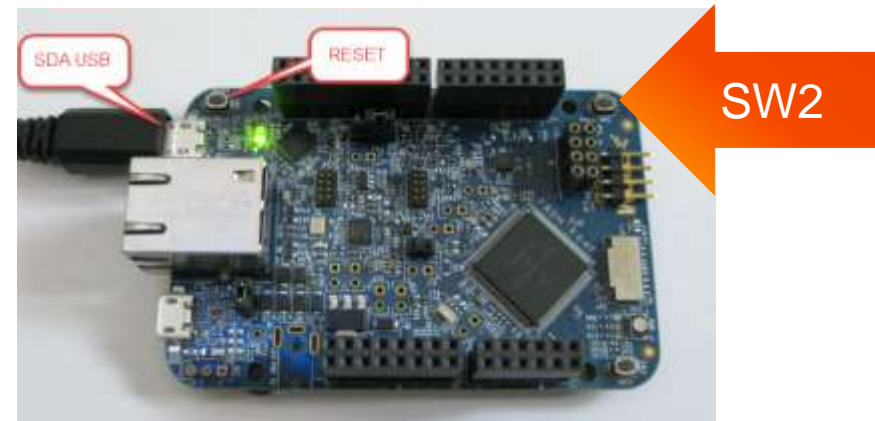
    (void) status;
    gcov_exit(); /* call coverage exit routine */
    /* turn on all LED's ==> WHITE */
    GPIO_DRV_ClearPinOutput(led_red);
    GPIO_DRV_ClearPinOutput(led_green);
    GPIO_DRV_ClearPinOutput(led_blue);
    for(;;){} /* does not return */
}

#endif /* COV_DO_COVERAGE */
```



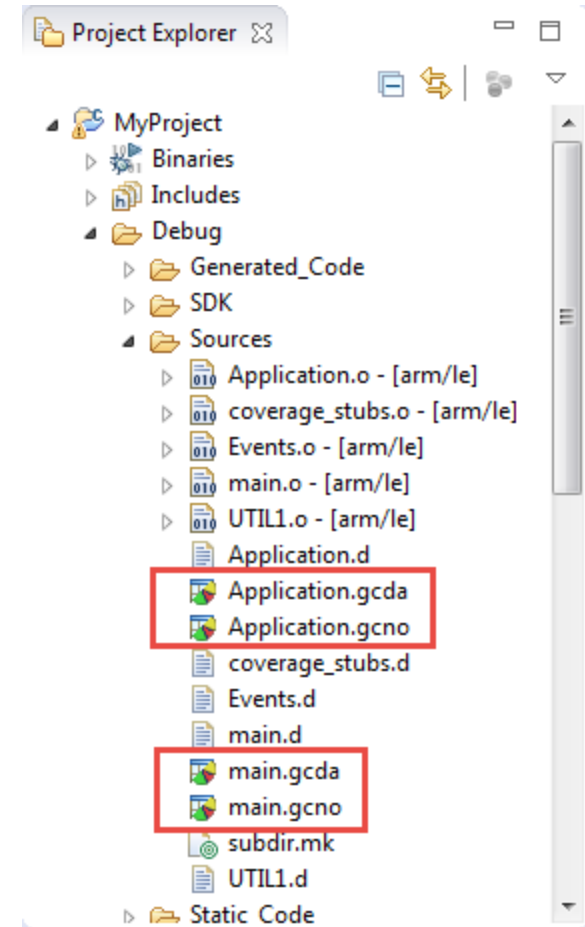
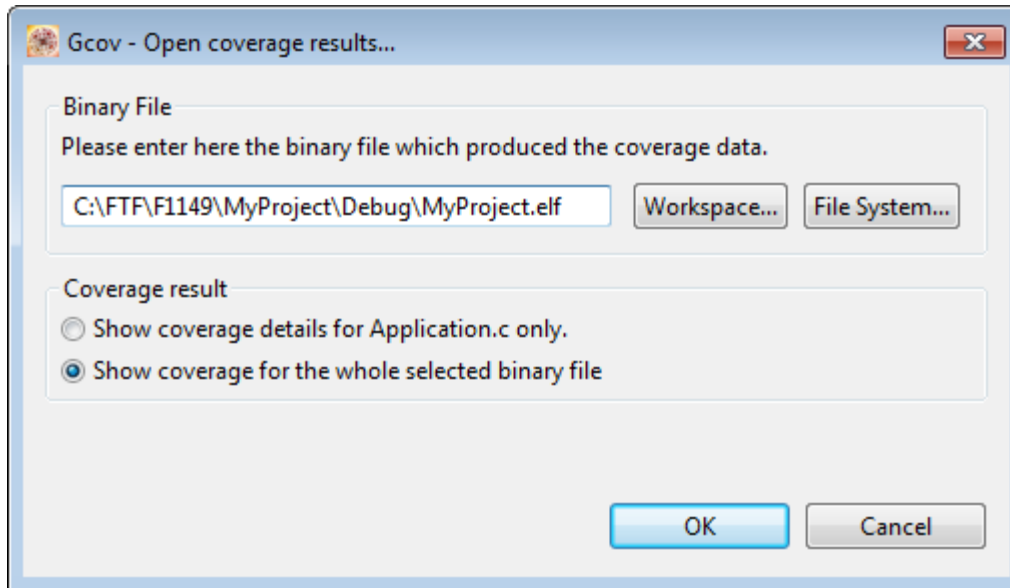
Lab: First Coverage Run

- Clean: delete 'Debug' folder in project
- Build application
- Debug application with **Segger**
- Run application
 - RED LED is blinking
- Press SW2
 - LED color changes
- After >5 SW2 presses
 - LED turned off
 - `_exit()` called by application
 - Trace dump written host
 - LED turns white after ~60 seconds
- Terminate debug session



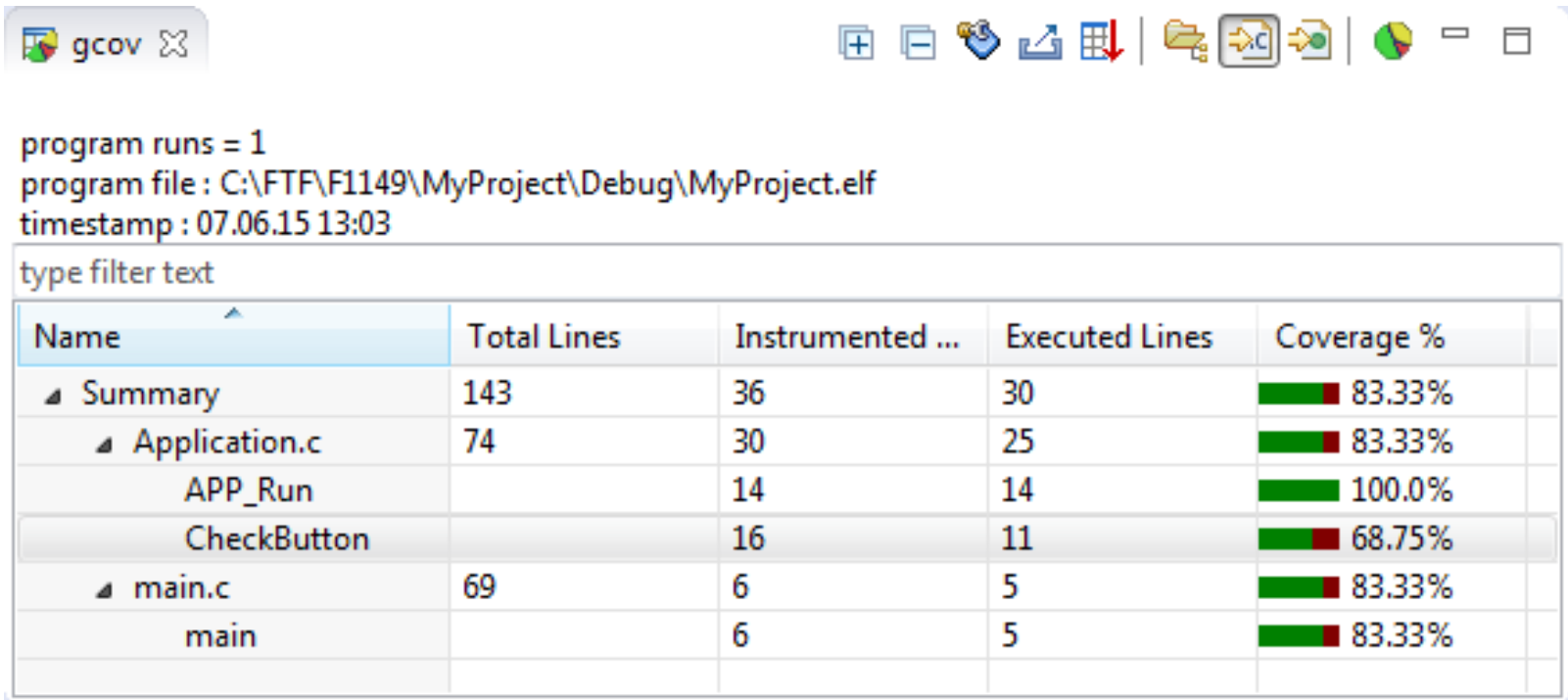
Open Data with Gcov

- Double-Click on any coverage file
 - Context menu Refresh
- Opens Dialog
- Specify application .elf file
- Press OK



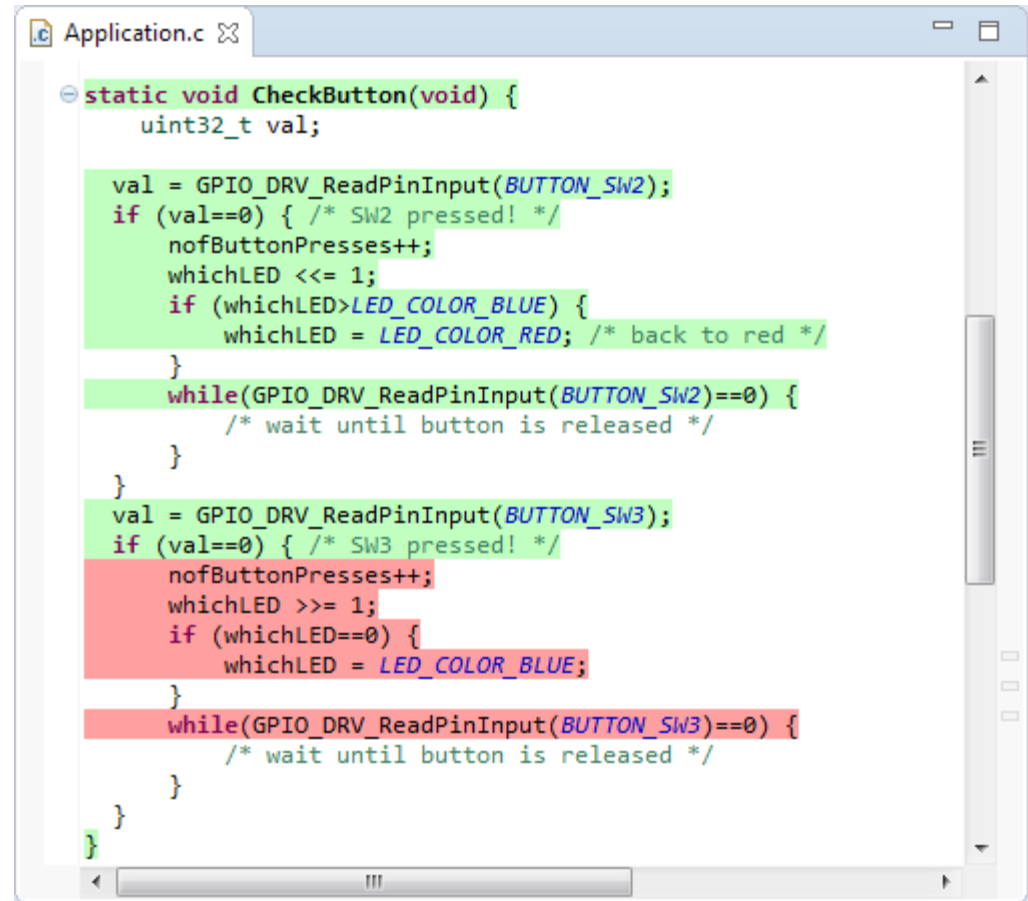
Gcov Eclipse View

- Gcov view showing status of coverage
 - summary, modules, functions
- Double-click on File/function to show details
 - Opens Source view



Coverage Information in Source View

- Green: executed/covered
- Red: not covered
- As only used SW2:
 - Code for SW3 not handled

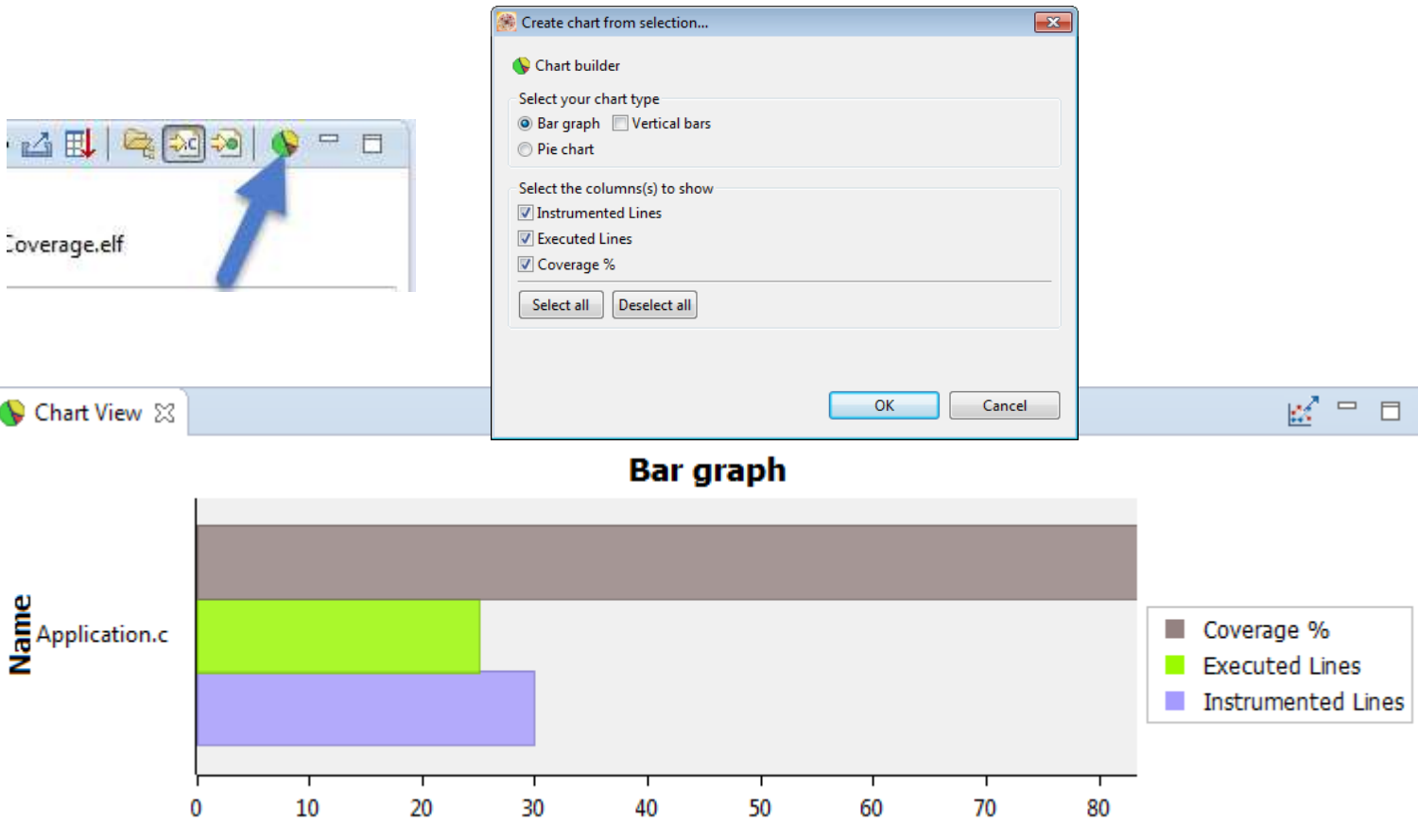


```
Application.c ✕
static void CheckButton(void) {
    uint32_t val;

    val = GPIO_DRV_ReadPinInput(BUTTON_SW2);
    if (val==0) { /* SW2 pressed! */
        nofButtonPresses++;
        whichLED <<= 1;
        if (whichLED>LED_COLOR_BLUE) {
            whichLED = LED_COLOR_RED; /* back to red */
        }
        while(GPIO_DRV_ReadPinInput(BUTTON_SW2)==0) {
            /* wait until button is released */
        }
    }
    val = GPIO_DRV_ReadPinInput(BUTTON_SW3);
    if (val==0) { /* SW3 pressed! */
        nofButtonPresses++;
        whichLED >>= 1;
        if (whichLED==0) {
            whichLED = LED_COLOR_BLUE;
        }
        while(GPIO_DRV_ReadPinInput(BUTTON_SW3)==0) {
            /* wait until button is released */
        }
    }
}
```

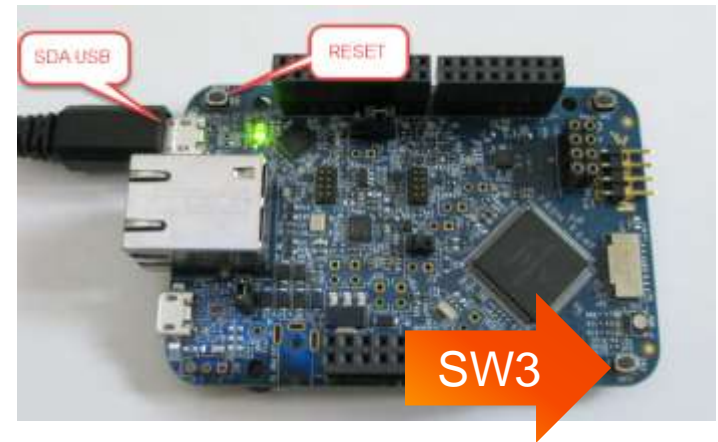
gcov Chart View

- Use 'chart' icon to open chart view
- Can save/export data in different formats



Multiple Coverage Runs

- Can run multiple test scenarios/coverage runs
- Gcov library will append information to files
 - Semihosting easily supports this 😊
- Run multiple sessions
 - This time, use SW3
 - Coverage should reach 100% 😊



Name	Total Lines	Instrumented ...	Executed Lines	Coverage %
Summary	143	36	35	97.22%
Application.c	74	30	30	100.0%
APP_Run		14	14	100.0%
CheckBoxon		16	16	100.0%
main.c	69	6	5	83.33%
main		6	5	83.33%

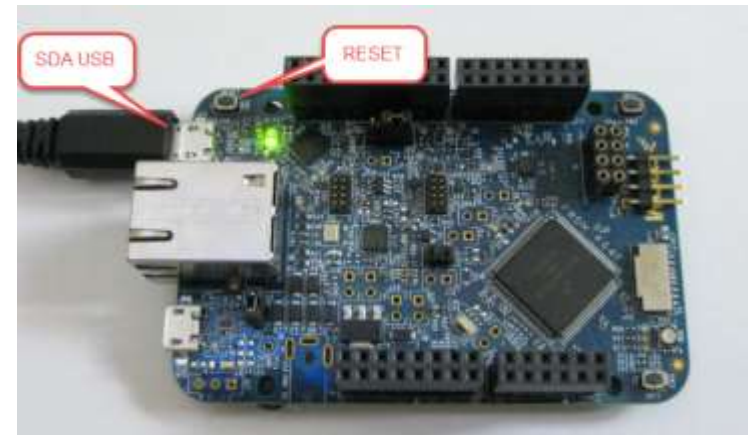
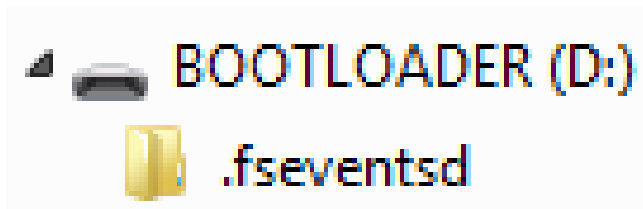
Summary

- Coverage for test coverage verification
- Generate Coverage information
 - Instrument code
 - Generate Instrumentation Information
 - Run and collect information
 - Analyze results with gcov
- Dump Coverage Information
 - Debugger file dump: not very user friendly
 - Semihosting: only with debugger attached, not very fast
 - Other options: SD card, custom communication channel, ...
- Further information:
 - <http://mcuoneclipse.com/2014/12/26/code-coverage-for-embedded-target-with-eclipse-gcc-and-gcov/>
 - <http://mcuoneclipse.com/2015/05/31/code-coverage-with-gcov-launchpad-tools-and-eclipse-kinetis-design-studio-v3-0-0/>
 - <http://mcuoneclipse.com/2014/11/01/illustrated-step-by-step-instructions-updating-the-freescale-freedom-board-firmware/>
 - <http://mcuoneclipse.com/2014/06/06/semihosting-with-kinetis-design-studio/>



WAIT! Load P&E FRDM-K64F OpenSDA Firmware

- Power Board with 'SDA USB' while RESET button pressed
 - Green LED near connector blinks slowly
 - Board enumerates as BOOTLOADER
 - Copy firmware file (**DEBUG-FRDM-K64F_Pemicro_v108a_for_OpenSDA_v2.0.bin**) to BOOTLOADER drive
 - Green LED blinks fast
- Repower board with SDA USB port normally
 - USB driver might enumerate
 - Green LED is on



Session Closing

What we have achieved...



Results

- Advanced Features with Kinetis Design Studio IDE
 - Version Control
 - Build automation
 - Debug automation
 - Code Coverage/Profiling



For Further Information

- Kinetis Design Studio IDE
 - www.freescale.com/kds
- Kinetis Software Development Kit (SDK)
 - www.freescale.com/ksdk
- FreeRTOS website
 - www.freertos.org
- Examples and Tutorials for Kinetis devices
 - www.mcuoneclipse.com
- Contact
 - Mark Ruthenbeck, mark.ruthenbeck@freescale.com
 - Prof. Erich Styger, erich.styger@freescale.com





www.Freescale.com

Backup Material

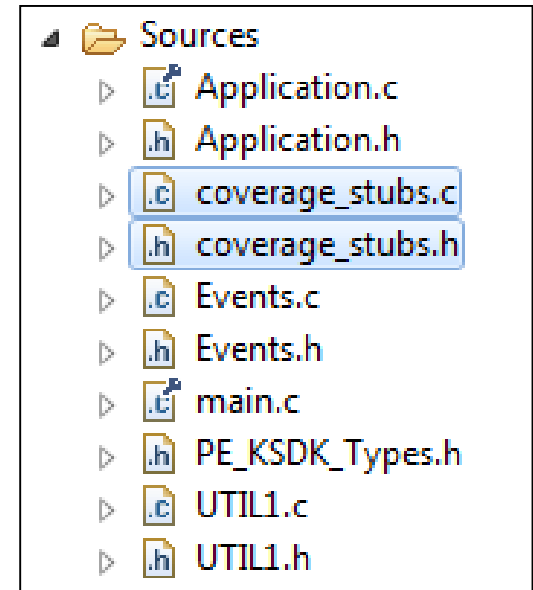


Dumping Coverage Data with GDB

Following material shows how to use GDB debugger commands to dump the coverage data to files on the host. This approach does not require semihosting. However, it does not allow combining multiple coverage runs without additional tools.

Coverage Stubs

- Set of coverage support routines
 - `_exit()`, `_write()`, `_open()`, `_close()`, ...
- Coverage will write 'file' with data
 - Use semihosting to write data
 - Can intercept `_write()` to dump data with gdb to file (See Backup material)



```
int _write(int file, char *ptr, int len) {
    static unsigned char gdb_cmd[128]; /* command line which can be used for gdb */
    (void)file;
    /* construct gdb command string */
    UTIL1_strcpy(gdb_cmd, sizeof(gdb_cmd), (unsigned char*)"dump binary memory ");
    UTIL1_strcat(gdb_cmd, sizeof(gdb_cmd), fileName);
    UTIL1_strcat(gdb_cmd, sizeof(gdb_cmd), (unsigned char*)" 0x");
    UTIL1_strcatNum32Hex(gdb_cmd, sizeof(gdb_cmd), (uint32_t)ptr);
    UTIL1_strcat(gdb_cmd, sizeof(gdb_cmd), (unsigned char*)" 0x");
    UTIL1_strcatNum32Hex(gdb_cmd, sizeof(gdb_cmd), (uint32_t)(ptr+len));
    return len; /* on success, return number of bytes written */
}
```

Exit Sequence: Writing Coverage Information

- Have a breakpoint set on `_write()` exit
- Debugger will stop when application calls `_exit()`

```
main.c coverage_stubs.c coverage_stubs.h coverage_stubs.c
static const unsigned char *fileName; /* file name use
int _write(int file, char *ptr, int len) {
    static unsigned char gdb_cmd[128]; /* command line w
    (void)file;
    /* construct gdb command string */
    UTIL1_strcpy(gdb_cmd, sizeof(gdb_cmd), (unsigned cha
    UTIL1_strcat(gdb_cmd, sizeof(gdb_cmd), fileName);
    UTIL1_strcat(gdb_cmd, sizeof(gdb_cmd), (unsigned cha
    UTIL1_strcatNum32Hex(gdb_cmd, sizeof(gdb_cmd), (uint32_t)ptr);
    UTIL1_strcat(gdb_cmd, sizeof(gdb_cmd), (unsigned char*)" 0x");
    UTIL1_strcatNum32Hex(gdb_cmd, sizeof(gdb_cmd), (uint32_t)(ptr+len));
    return len; /* on success, return number of bytes written */
}

main.c
int main(void)
{
    int j;
    /* Write your code here */
    static_init();
    /* This for loop should be replaced. By default this
    for (j=0;j<5;j++) {
        i+=COV_Demo();
    }
    #if DO_COVERAGE
    _exit(0); /* write coverage information */
    #endif
    for(;;) {}
    /* Never leave main
    return 0;
```


Construction of Dump Command

- Hover over `gdb_cmd` to show pop-up window to show variable
- Click into pop-up and copy command to clip board (CTRL+C)

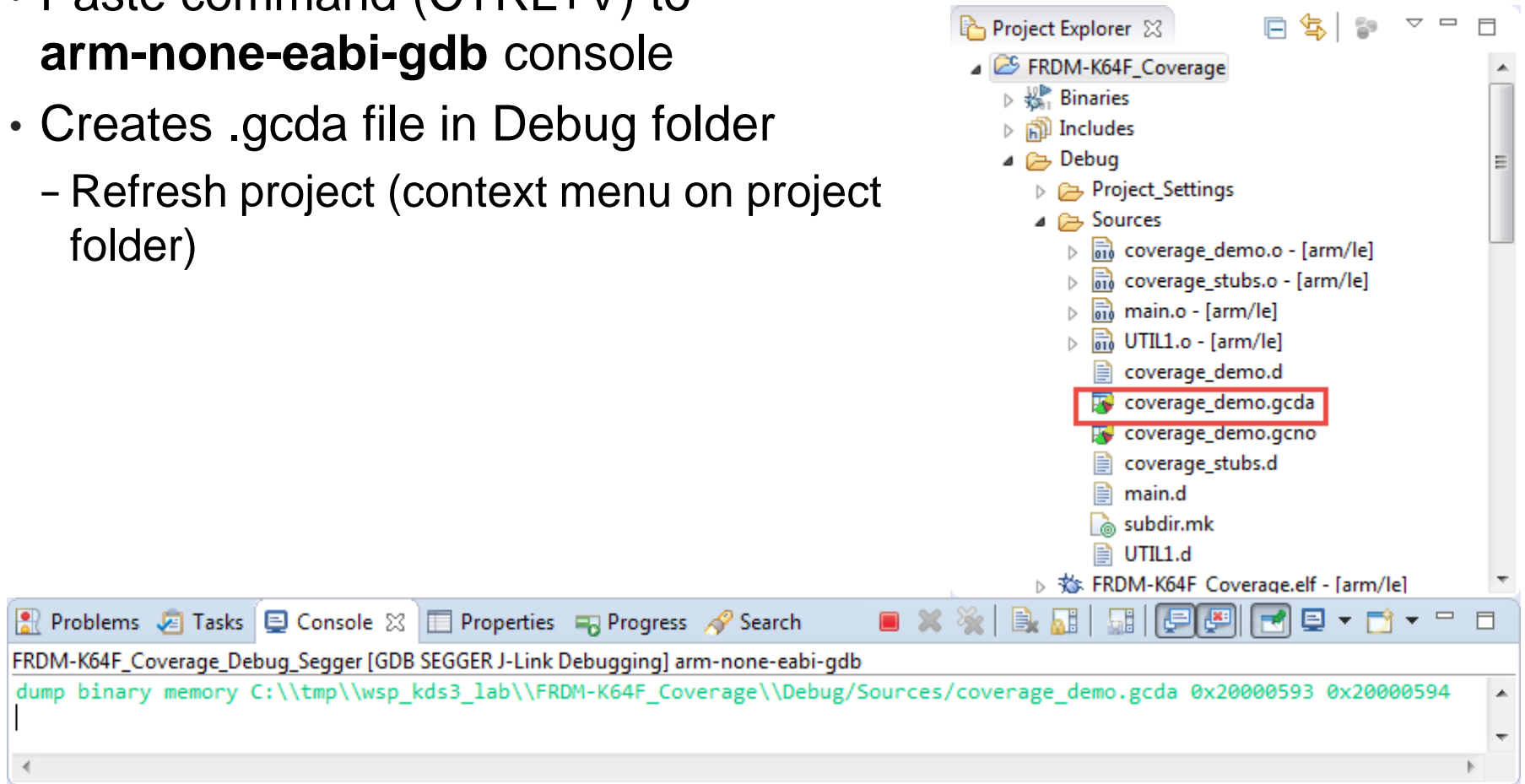
The screenshot shows an IDE with three main components:

- Code Editor:** Displays C code in `coverage_stubs.c`. The `return` statement is highlighted in green, and a red arrow points to the `gdb_cmd` variable.
- Variable Inspection Window:** A table showing the value of `gdb_cmd` at the current execution point.

Expression	Type	Value
<code>gdb_cmd</code>	unsigned char [128]	0x1fff0144
[0...99]	unsigned char [100]	0x1fff0144
[100...127]	unsigned char [28]	0x1fff01a8
- Console:** Shows the output of a build process, including the command `make -j8 all` and the invocation of the cross-compiler.

Saving File

- Paste command (CTRL+V) to **arm-none-eabi-gdb** console
- Creates .gcda file in Debug folder
 - Refresh project (context menu on project folder)





www.Freescale.com