Getting Started with Green Hills Tools & Multicore Qorivva MCUs

AMF-AUT-T1059

Steve Mihalik
Freescale Senior Field Application Engineer

Gil Lauer
Green Hills Senior Field Application Engineer

September 2013

Agenda

- Hands-on #1: Create a MULTI project using default MULTI wizard that includes a minimal flash-based program
- Hands-on #2: Create a minimal ram-based program in the same project
- Hands-on #3: Create a program that imports application source code for each core which implements interrupts
- Hands-on #4: Download program to Freescale evaluation board and perform basic debug
- Hands-on #5: Use advanced development tools on some example programs

Qorivva MPC5746M microcontrollers are used for hand-on exercises in this presentation.

Session Introduction

- This hands-on session walks through creating and debugging software projects using Green Hills tools for single core and multicore Qorivva microcontrollers.
- These tools and microcontrollers are winning designs and growing in popularity.
- Getting started with new tools is always a challenge. This session provides a controlled environment to streamline the learning process.
- The presenters have years experience helping customers get started with these tools and microcontrollers.
- The session focuses first on building projects from scratch with existing code. The second part walks through common debug steps and demonstrates advanced tool capabilities.

Session Objectives

- After completing this session you will be able to use Green Hills tools with Freescale microcontrollers to:
  - Create from scratch a MULTI project with flash and RAM based single & multicore programs
  - Identify required initializations in those programs
  - Create a multi core program that imports external application code
  - Download and perform basic debug operations on a created program
  - Apply advanced development tools including trace, stack analysis, run-time error checking

Record Breaking EEMBC Results - AutoBench

Green Hills SOFTWARE

Green Hills **10% - 30%** faster than other compilers

EEMBC™

	Processor	Compiler	Mark™	Energymark™	Type	Certified	Notes
	Freescale MPC5566 - 144MHz	Green Hills: PPC Version 5.20	121.0		OTB	01/14/11	
	Freescale MPC5607B - 64MHz	Green Hills: PPC Version 5.20	27.3		OTB	01/14/11	
	Freescale MPC5644A - 150MHz	Green Hills: PPC Version 5.20	149.6		OTB	01/13/11	
	Freescale MPC5674F - 264MHz	Green Hills: PPC Version 5.20	305.1		OTB	01/11/11	
	Freescale MPC8641D 1.5G	Green Hills Multi V4.2.3	1612.7		OTB	11/23/06	Single core used for benchmarking
	Freescale MC8548 - 1333MHz	Green Hills MULTI v4.2.3	1289.2		OTB	10/23/06	
	Freescale i.MX21 - 266 MHz	ARM-Linux-gcc-3.3.2-glibc-2.3.2	29.6		OTB	08/26/05	
	Freescale i.MX31-532MHz	armv6fl-montavista-linuxeabi-gcc 3.43	126.6		OTB	08/26/05	

Optimizing compilers achieves record certified EEMBC scores for automotive

see http://www.eembc.org/benchmark/index.php?suite=ALL&type=PRO&member_seq=206

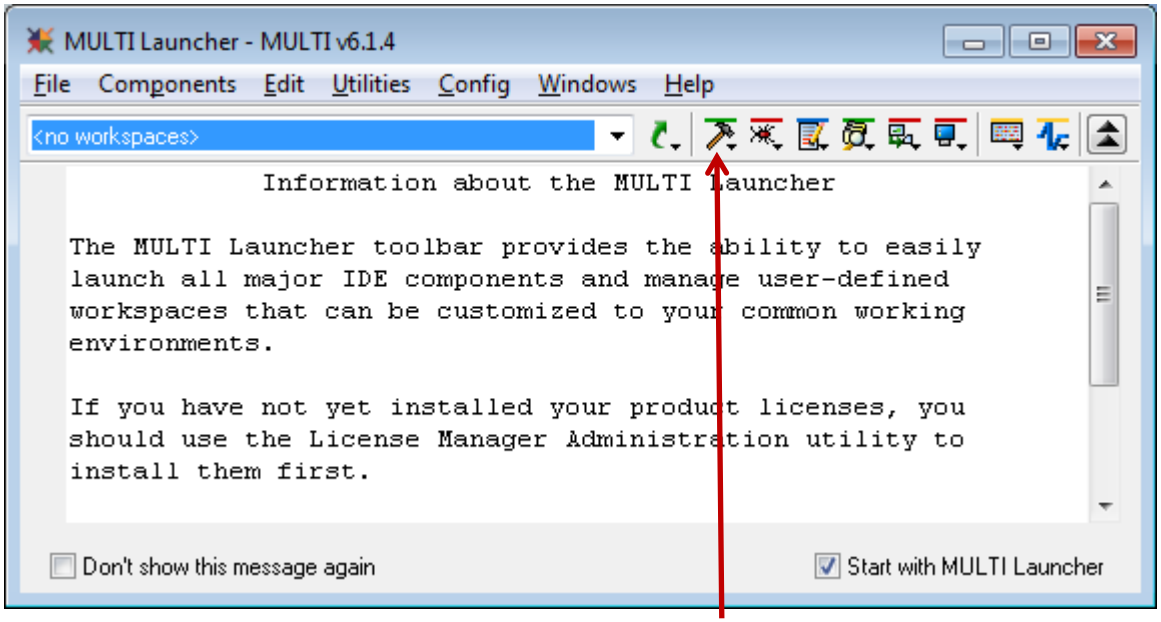
freescale™

Hands-on 1:

- Create a MULTI project that includes a minimal flash-based program

Hands-on 1.1: **Create Project**

- Start MULTI. The MULTI Launcher screen appears:



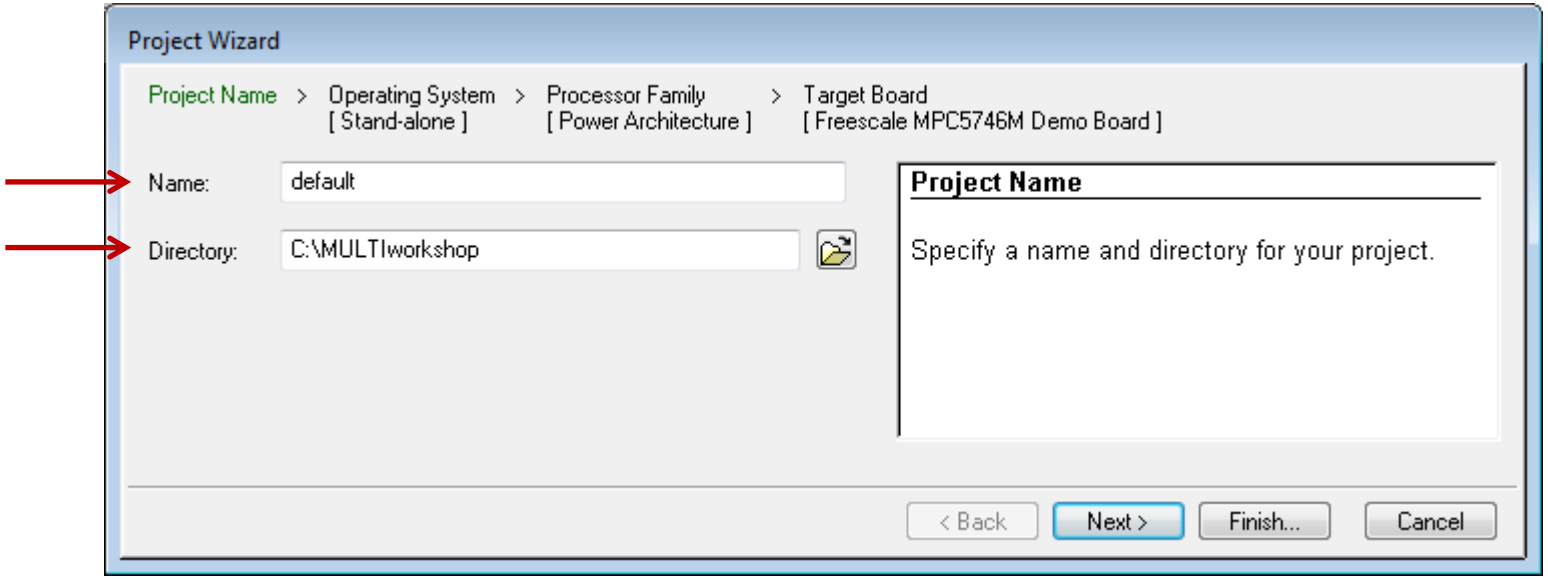
- Start MULTI's Project Wizard: click on **project manager icon** (hammer) and select "**Create Project**.."
  - Alternate action: File menu -> Create New Project…

Hands-on 1.2: **Top Level Project & Directory**

- **Name**: leave as **default** (common use)
- **Directory**: Type in Directory box to create new path and folder

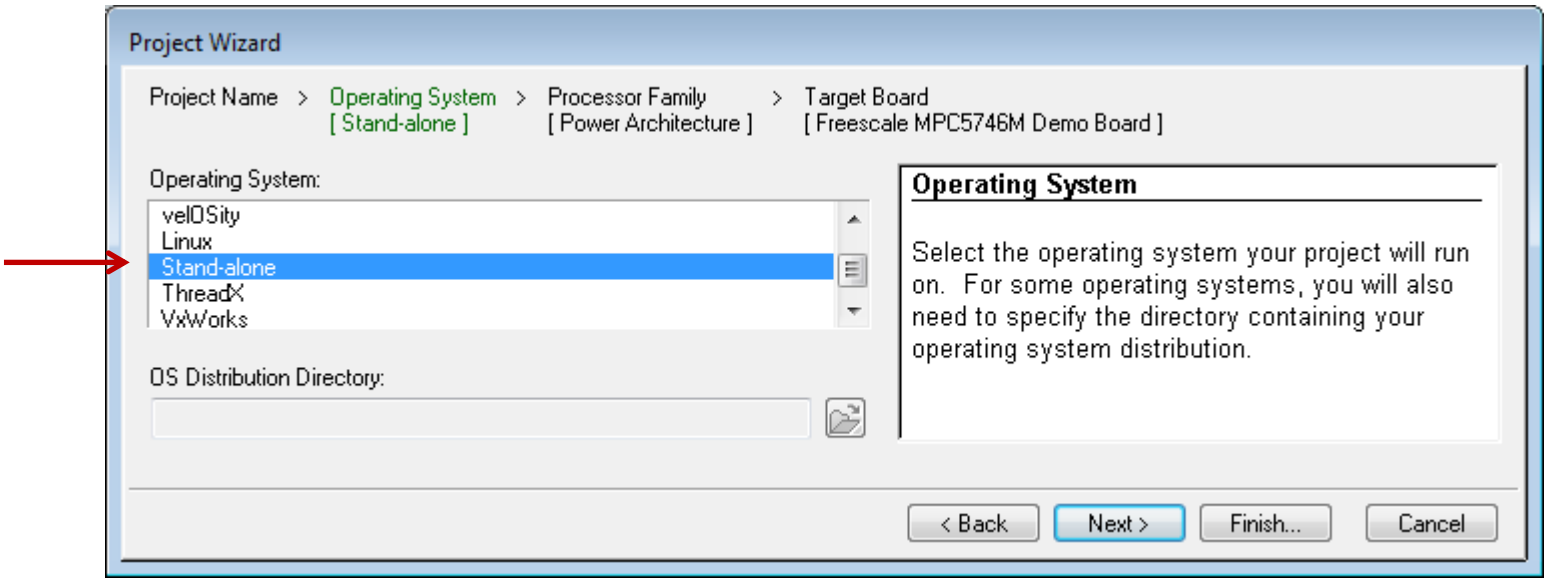
C:\MULTIworkshop

- Click **Next**



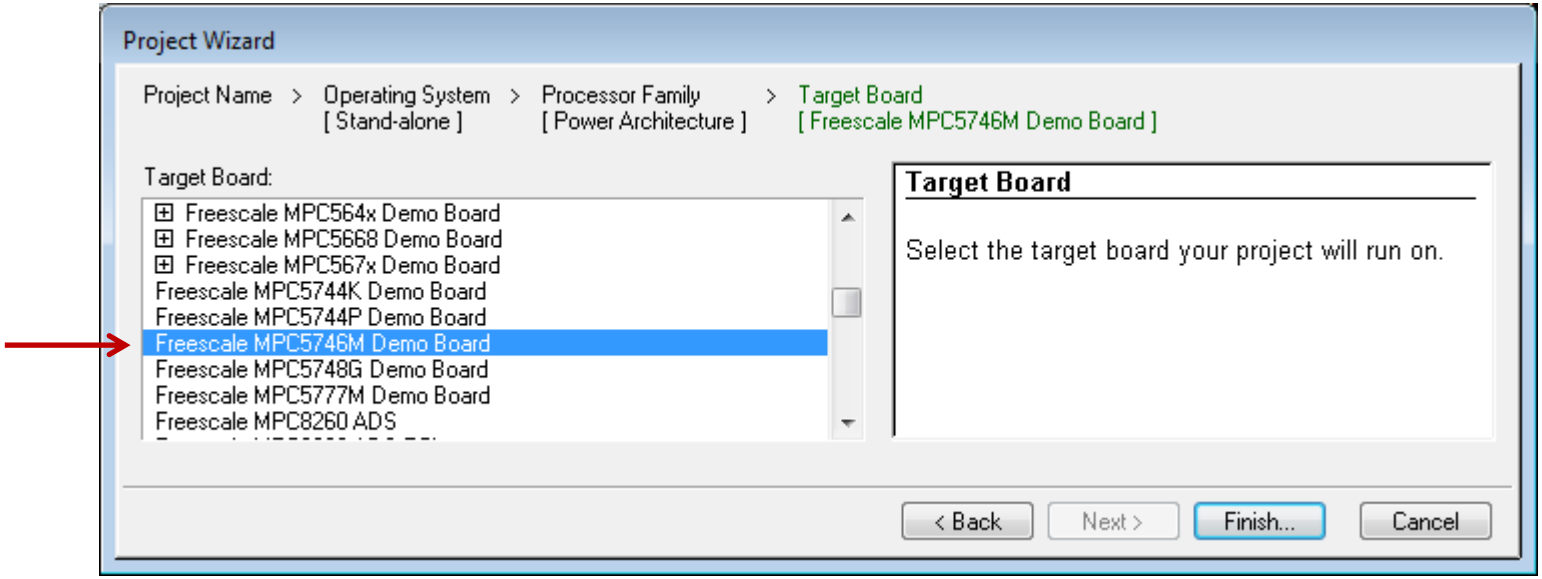
Hands-on 1.3: **Select Operating System**

- If not already highlighted, click on **Standalone** to select the operating system
- Click **Next**



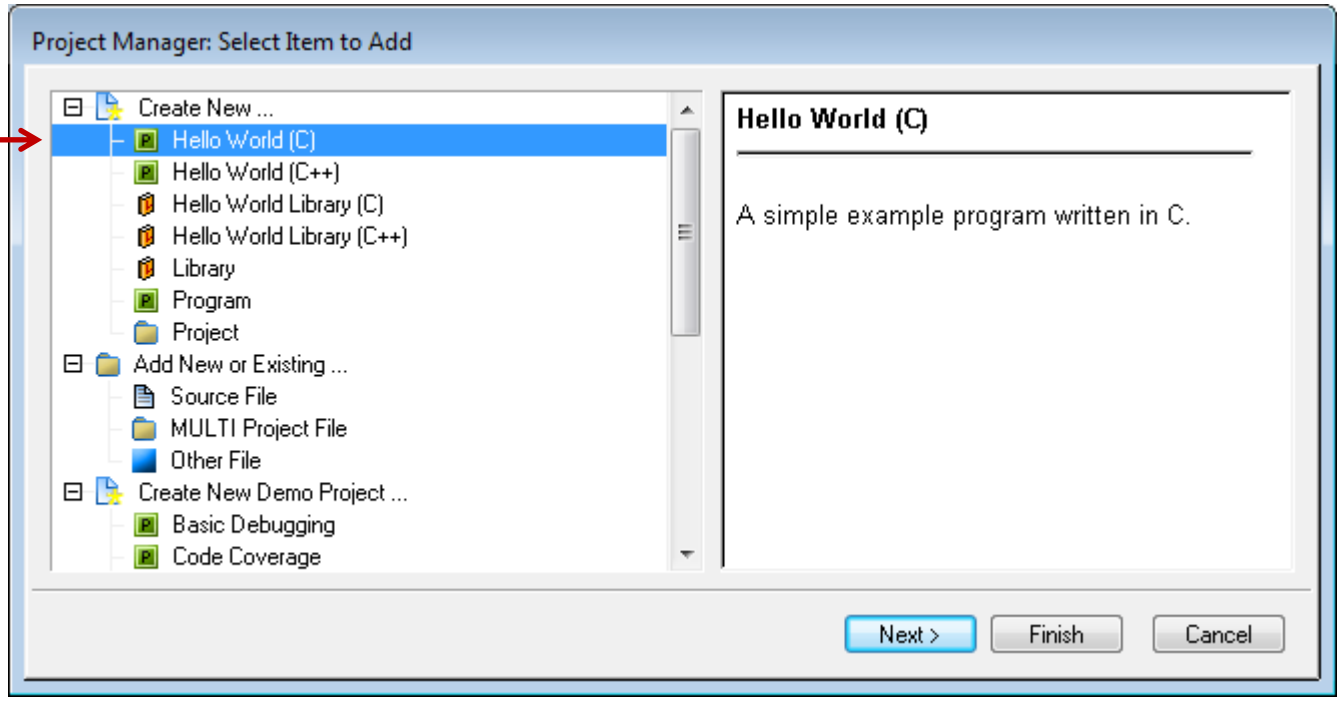
Hands-on 1.4: **Select Target Board**

- Select your target board: **Freescale MPC5746M Demo Board**
- Click **Finish**
  - A **warning message** may appear concerning memory size for Standard C++ executables. Click **OK** to exit the message.
  - **Top Project Created** message appears. Click **OK**



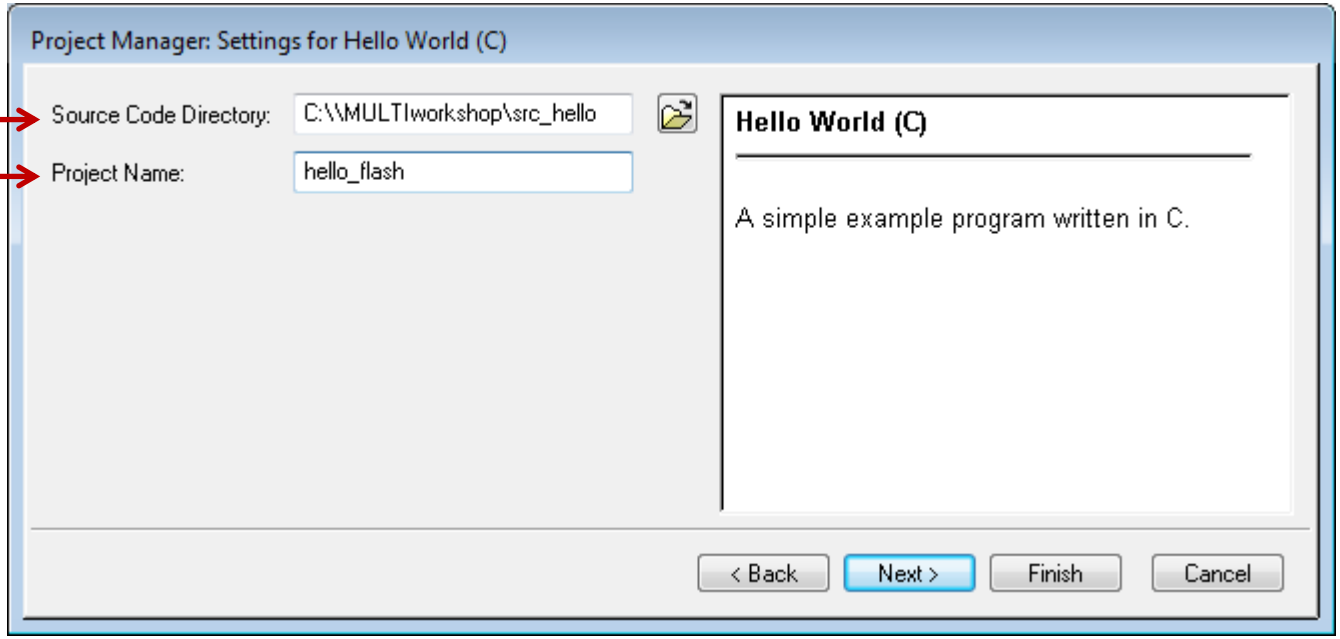
Hands-on 1.5: **Add Item(s) to Project**

- Two windows appear: the empty project and an “add” window over it
- Add desired item(s) to the project. In this example, keep **Hello World (C)** selected
- When the file is selected, click on **Next**



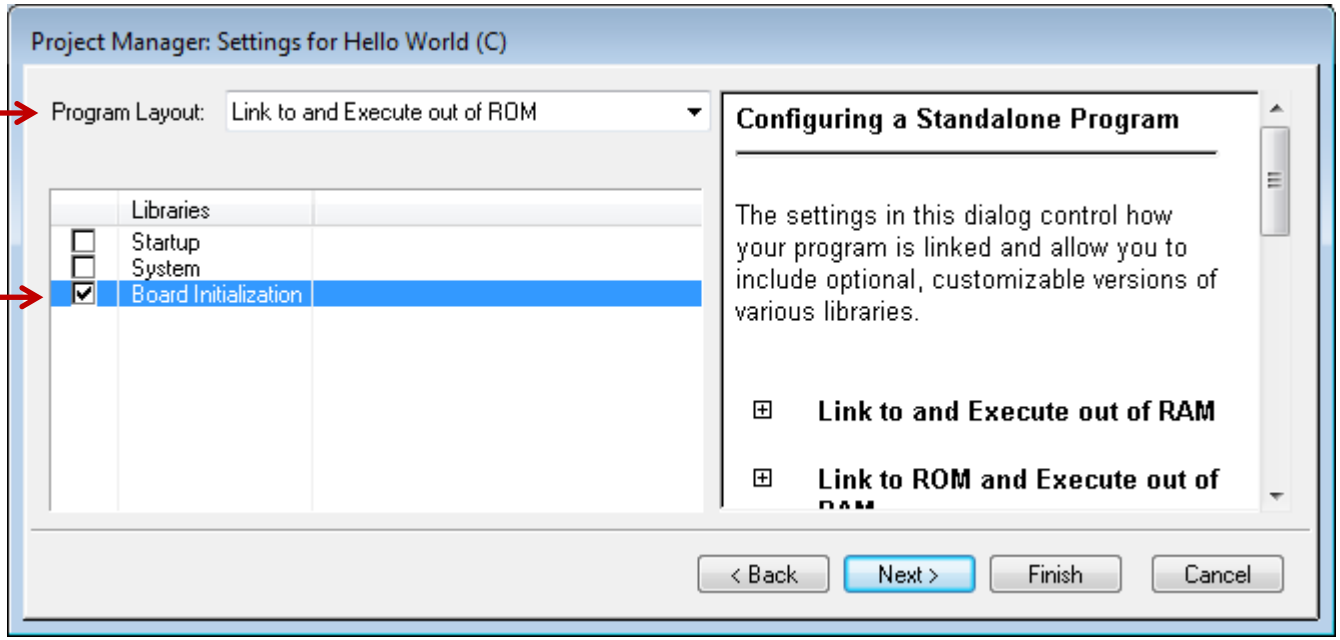
Hands-on 1.6: **Specify Directory and Name**

- **Source Code Directory**: use default **C:\MULTIworkshop\src_hello**
- **Project Name**: change from hello to **hello_flash** as below
  - This later appears as Program Name
- Click **Next**



Hands-on 1.7: **Configure Linker and Libraries**

- **Program Layout**: Select "**Link to and Execute out of ROM**"
- **Libraries**: Select "**Board Initialization**" to run from flash
- Click **Next**



Hands-on 1.8: **Generating VLE Instructions Note**

Some earlier Qorriva MCUs support both full 32-bit (Book E) and a variable instruction length (VLE) of 16- & 32-bit instructions.

MPC5746M only supports VLE instructions, which will be the default option by the MULTI wizard so **no action is required**.

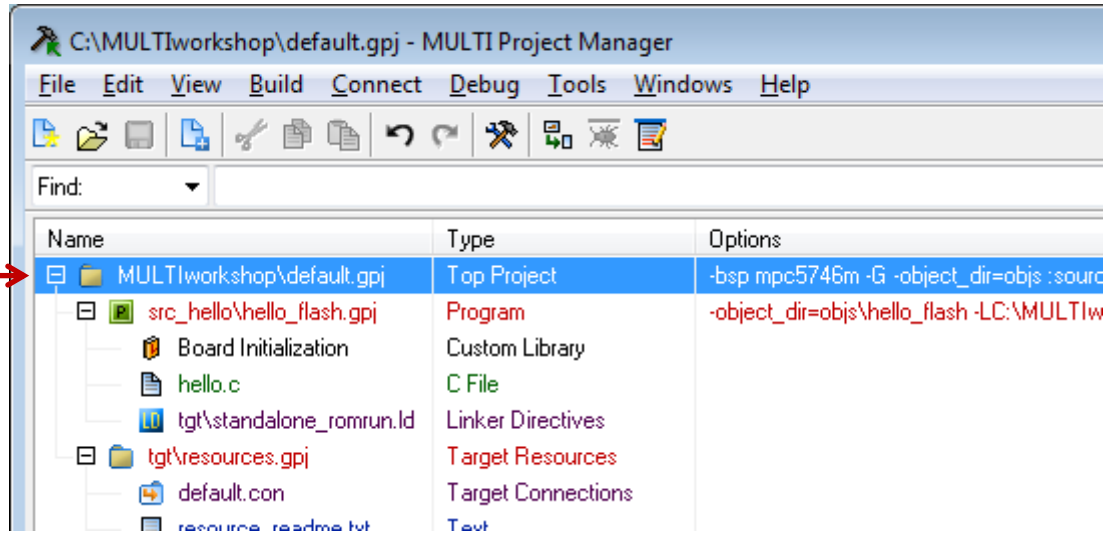
Note: If the MCU supports both Book E and VLE instructions, the default will be Book E instructions. If VLE instructions are desired, then set the project global settings:

1. Select **top project** & **right click** on it
2. Select "**Set Build Options**…"
3. Select the **All Options** tab
4. Expand "**Target**"
5. Select "**Instruction Set**"
6. Right click on "**VLE Code Generation and Libraries**"
7. Select "**On**"
8. Click **DONE**

Hands-on 1.9: **Debugger Compatibility - 1**

Various global project settings are often needed. One is made here for interoperability.

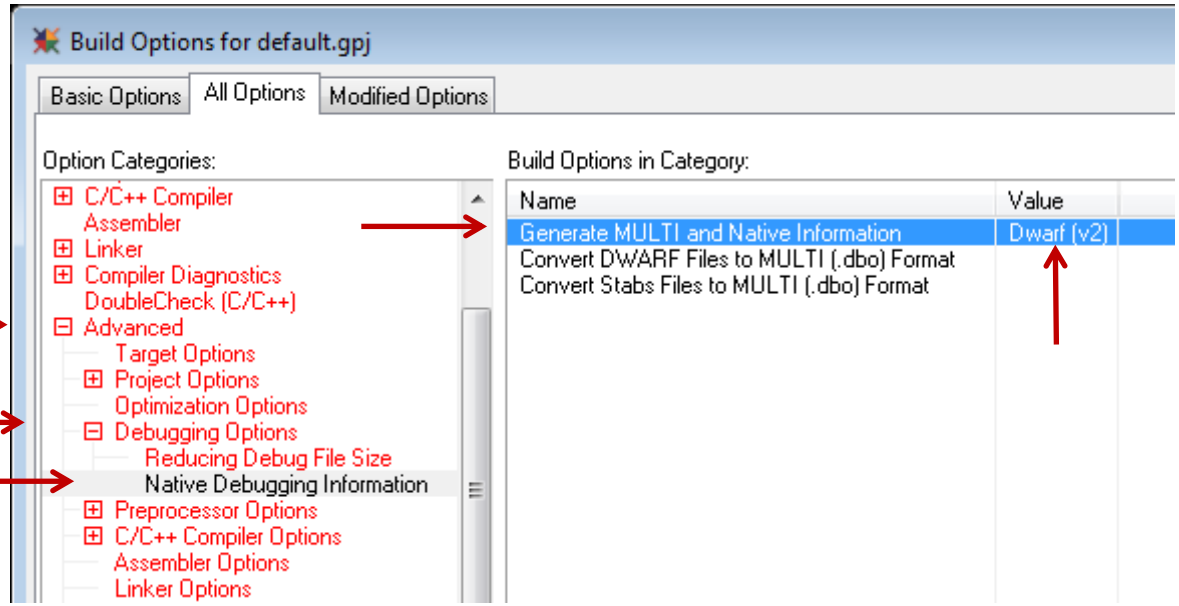
- First, **select top project** & **right click** on it
- Select "**Set Build Options**…"



Hands-on 1.9: **Debugger Compatibility - 2**

For compatibility with other debuggers, make all the project's programs generate DWARF output files & add .elf extension their names. Stay on the All Options tab

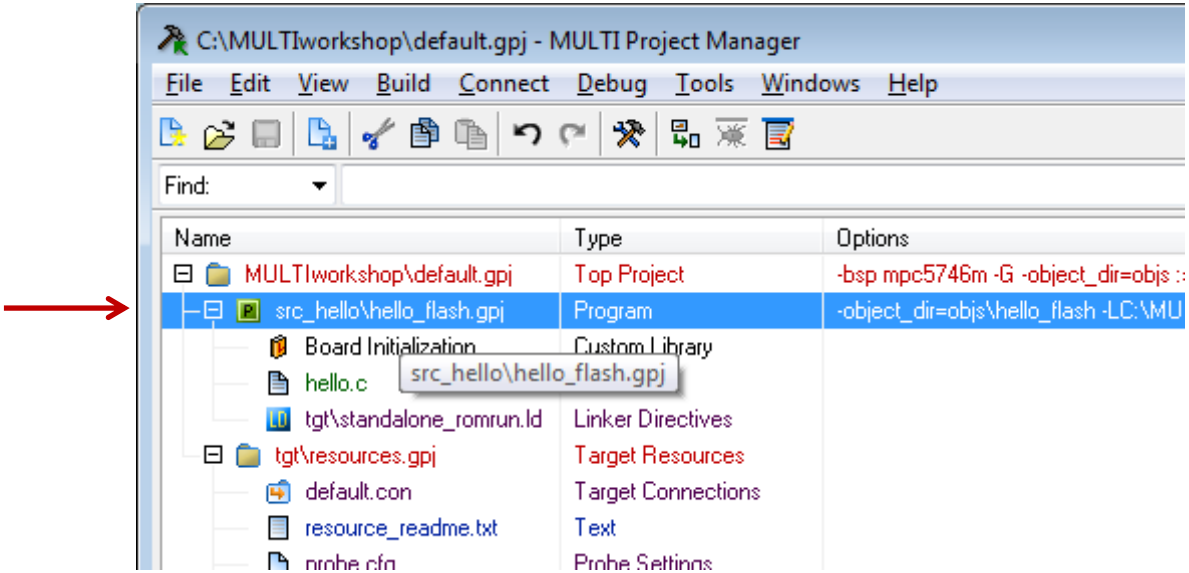
- Expand "**Advanced**"
- Expand "**Debugging Options**"
- Select "**Native Debugging Information**"
- Right click on "**Generate MULTI and Native Information**"
- Select "**Dwarf (v2)**"
- Click on "**OK**"
- Click on "**Done**" button in bottom right corner



Hands-on 1.9: **Debugger Compatibility - 3**

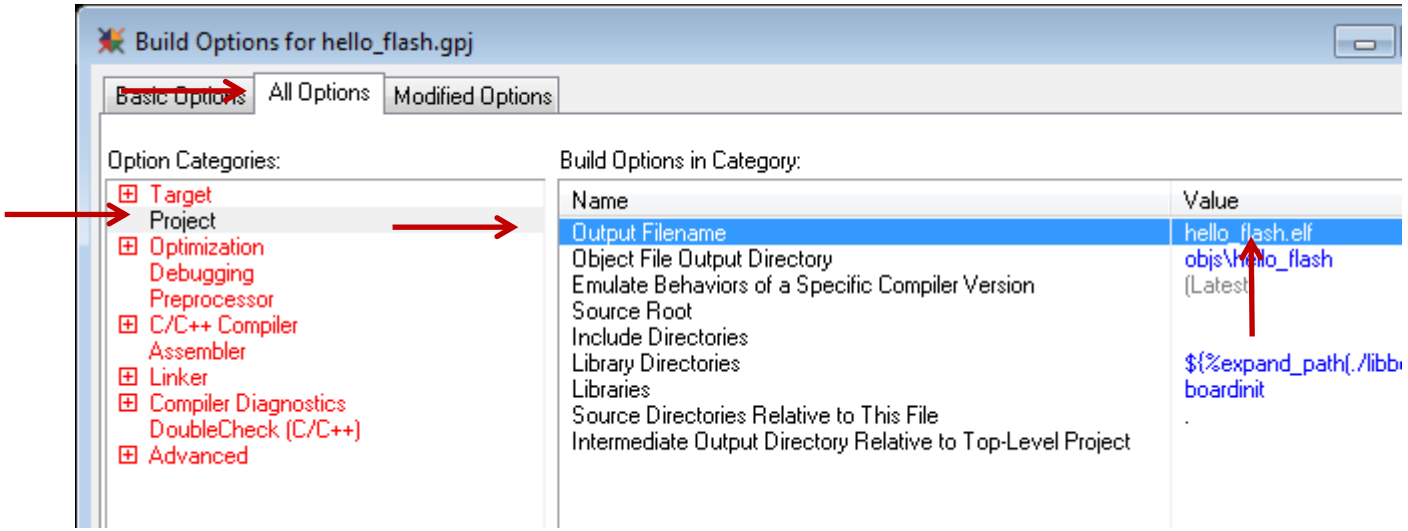
Change object file name to have an .elf extension because some it is required for some debuggers:

- In Project Manager window, **right click on the Program** we created and select "**Set Build Options**…"



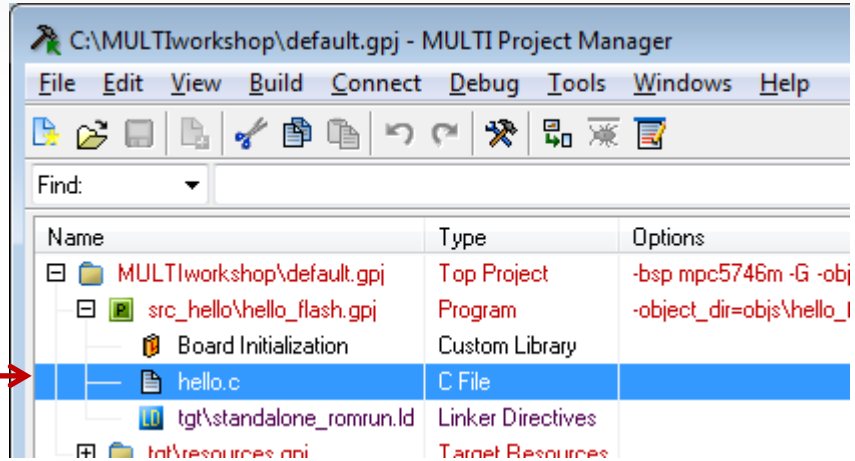
Hands-on 1.9: **Debugger Compatibility - 4**

- Select "**All Options**" tab
- Select "**Project**"
- Right click on "**Output Filename**"
- Select "**Set Output Filename**"
- Enter new filename with desired extension: "**hello_flash.elf**" & click **OK**
- Click "**Done**" button

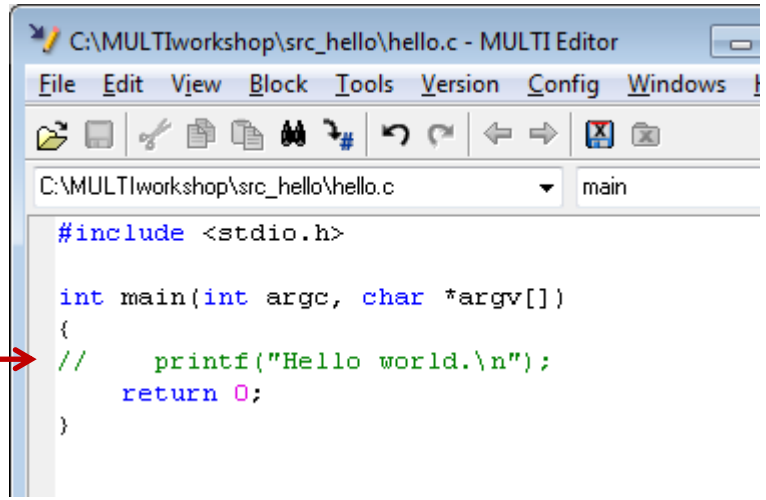


Hands-on 1.10: **Modify a Source File**

- **Double click** on “**hello.c**”

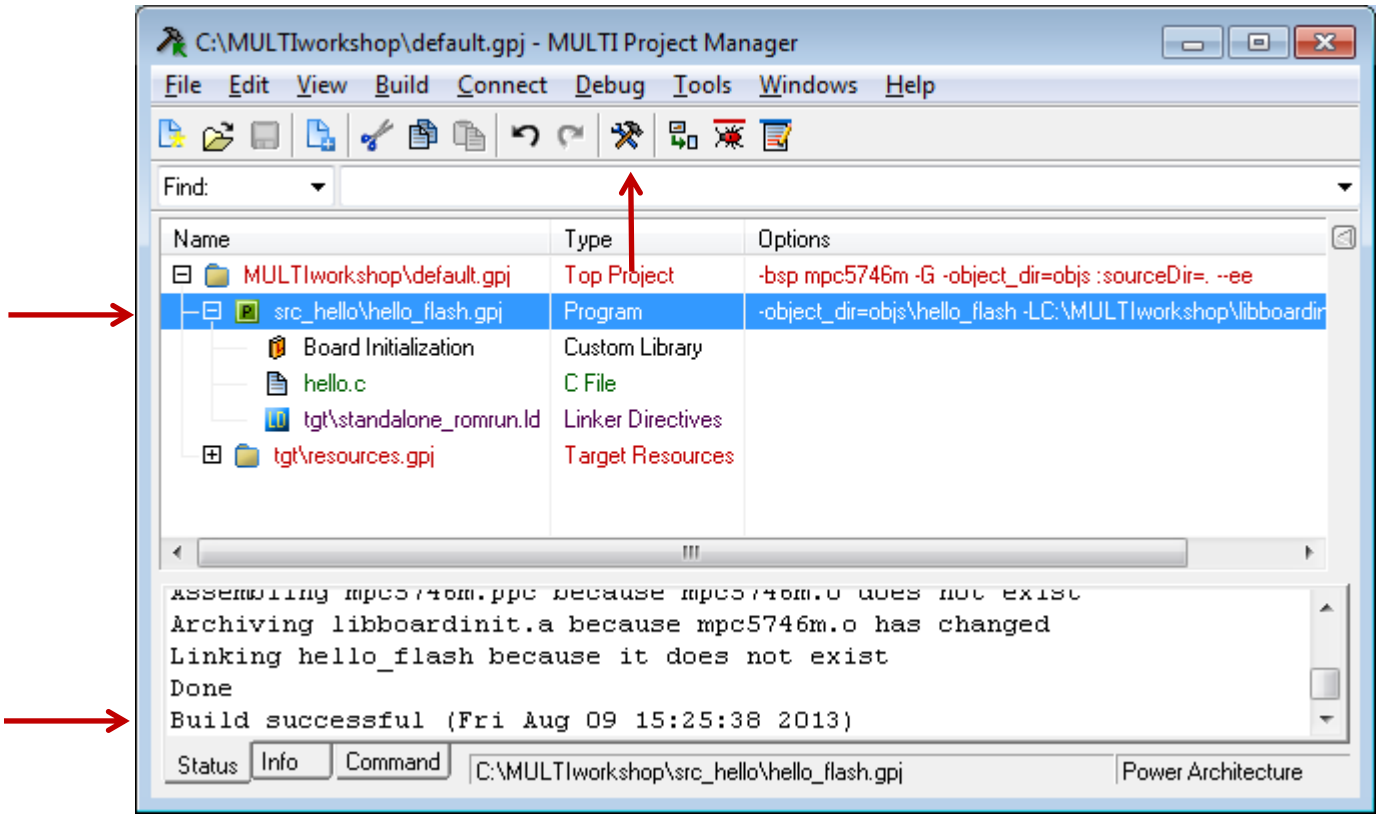


- **Comment out printf** statement as shown
- **Save and close** editor



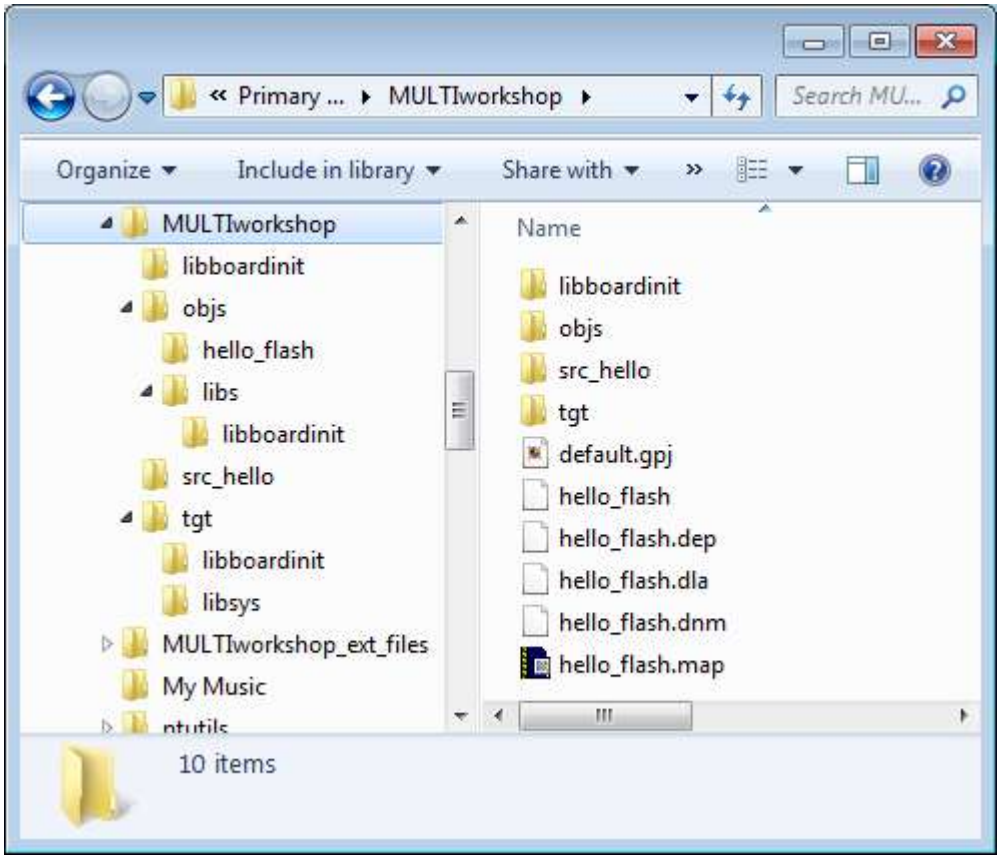
Hands-on 1.11: **Build Project**

- **Select hello_flash** program
- Click on **Build icon** (crossed tools)
- After program builds, Status tab shows Build successful



Hands-on 1: **Project Directories and Files**

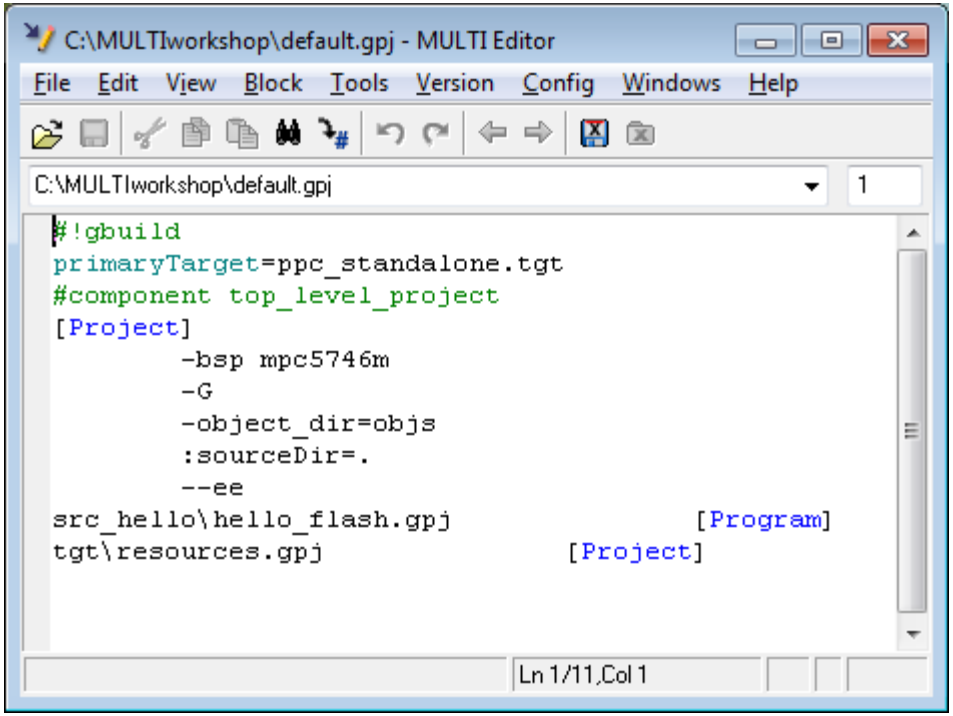
The built project has folders and files as shown below



Hands-on 1: **MULTI Tip - Editing Project File**

- The project file, default.gpj, is text and can be viewed edited with the MULTI editor or one like Wordpad
  - Example reason to edit: change a path name

- Example:
  - Select Top Project
  - Right click on it
  - Click on “Edit”
  - Screen at right appears that can be edited.



Hands-on 1: **Summary – Create Project, 1st Program**

Step			hello_flash target
Create Project (Project Wizard)	1	Create project	Create from MULTI Launcher
	2	• Top level project name • Select project directory	• default • C:\MULTIworkshop
	3	Select operating system	Stand-alone
	4	Select target board	MPC5674F
Create First Program - minimal flash (Project Wizard)	5	Add desired source to project	hello.c
	6	• Specify source directory • Specify program name	• C:\MULTIworkshop\hello_src • hello_flash
	7	• Configure linker • Select libraries	• Link & execute from ROM • Board init
Modify Parameters & Build Program (Program Manager)	8	If needed, specify instruction set	VLE used (16 & 32 bit instr.)
	9	Debugger compatibility	• Project: DWARF 2.0 • Program: .elf output filename ext'n
	10	Source file modification	Commented a line
	11	Build program	Build

Hands-on 1: **Qorivva Startup**

- Mandatory Startup
  1. Boot Header or Reset Configuration Half Word (RCHW) in flash
     - Determines reset configuration after reset, boot core is selected
     - Identifies start vector for boot core
  2. SRAM ECC initialization
  3. Some devices: Memory Management Unit (MMU) initialization
  4. Some devices: If using lockstep, write to core registers not initialized on power on reset
- Other initialization items (not on all devices) such as:
  - Mode Entry module
  - PLL
  - Watchdog servicing or disabling
  - Cache

Hands-on 1: **Startup – Boot Header**

- Boot Header is read from the first of several specific addresses in flash that contains a Boot Identifier = 0x5A

- Example: MPC5746M boot header configuration

  MPC5746M boot header structure
  - CPU2 is the normal boot core on MPC5746M
  - GHS puts boot header in mpc5746m.ppc

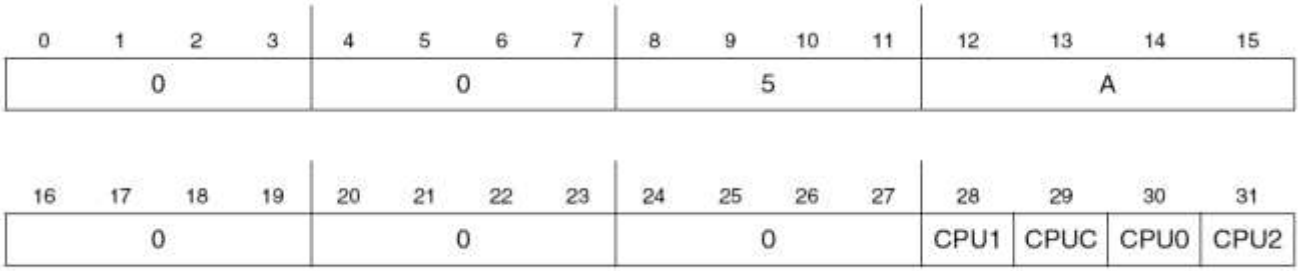


Figure 1236. Boot header configuration

Table 1185. Boot Header Structure

Address Offset	Contents
00h	Boot Header Configuration (see Figure 1236)
04h	CPU2 Reset Vector
08h	Configuration Bits
0Ch	Configuration Bits
10h	CPU0 Reset Vector
14h	CPU1 Reset Vector
18h	CPUC Reset Vector

Hands-on 1: **Startup - SRAM Initialization**

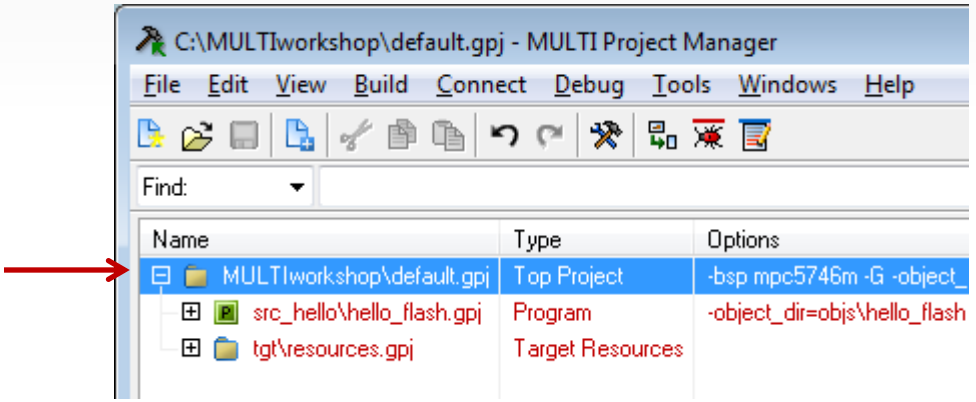
- SRAM must be initialized before it can be used, otherwise an ECC error is generated
- Initialization consists of writing any values to all SRAM locations, writing the full bus width
  - For example, if SRAM bus is 64 bits wide, do 64-bit writes
- This is done in assembly language after reset along with typical compiler initialization such as stack initialization, ROM copy, etc.
- Example SRAM initialization code is usually in the SRAM chapter of the device reference manual
- Examples:
  - Flash target - MULTI's resource file mpc5746m.pcc: Contains 6 lines of assembly language to loop through writing all of SRAM
  - RAM target – GHS Probe firmware initializes a configurable amount of SRAM

Hands-on 2:

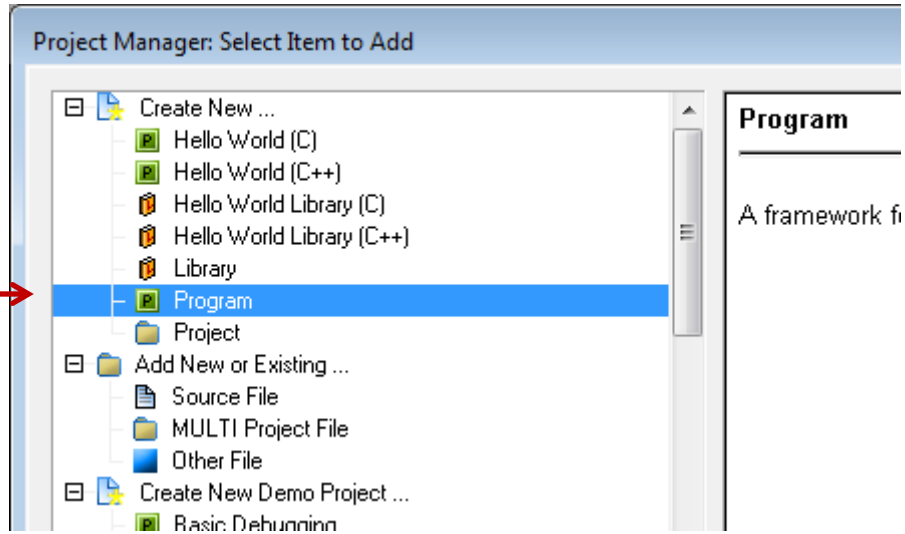
- Create a minimal ram-based program in the same project

Hands-on 2.1: **Add Program to Top Project**

- **Right click** on **Top Project**, then select "**Add Item into default.gpj**…"

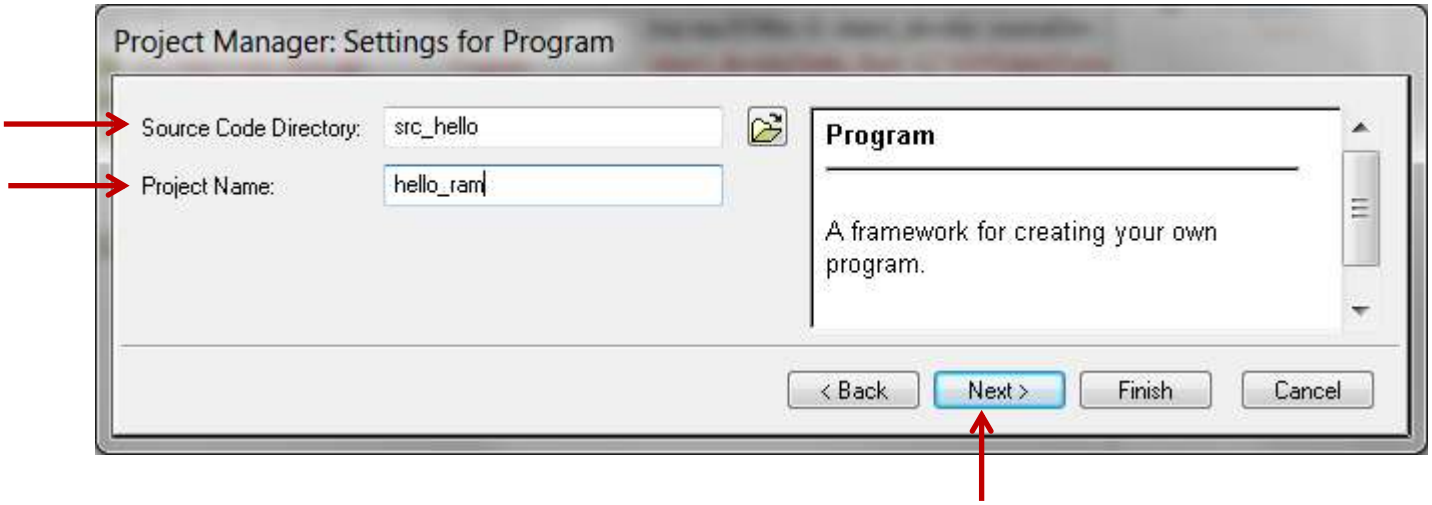


- Select **Program** in the Create New options
- Click **Next**



Hands-on 2.2: **Specify Directory and Name**

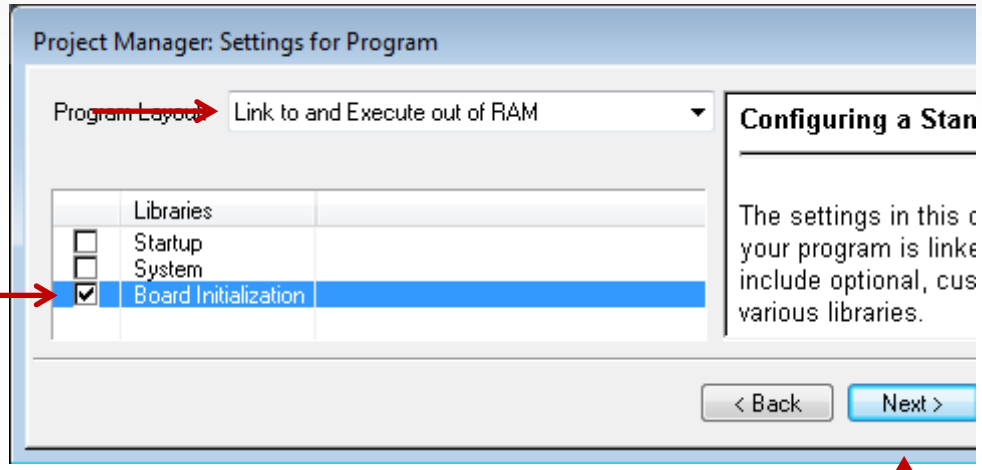
- **Source Code Directory**: change to use same as before: **src_hello**
  - **type it in so project will use a relative path**
- **Project Name**: change to **hello_ram**
- Click **Next**

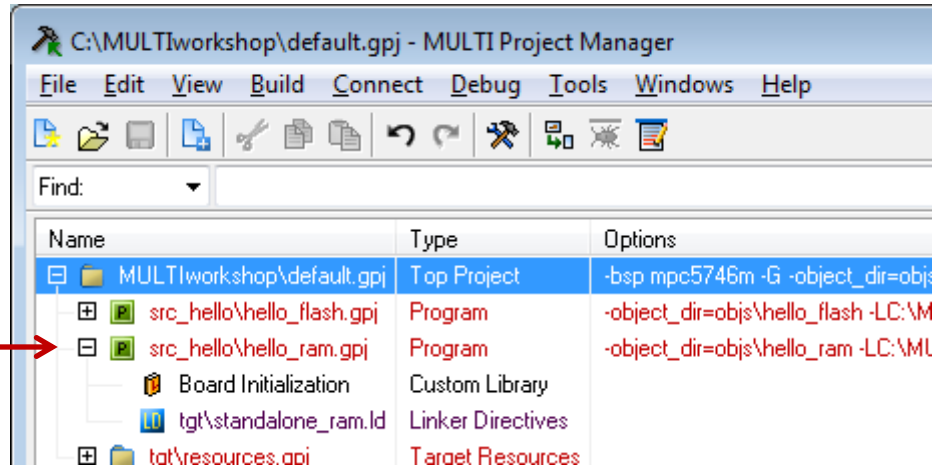


Hands-on 2.3: **Specify Link and Libraries**

- **Program Layout**: Select "**Link to and execute out of RAM**"
- **Libraries**: Select "**Board Initialization**" which has routines used to bring up cores.
- Click **Next**

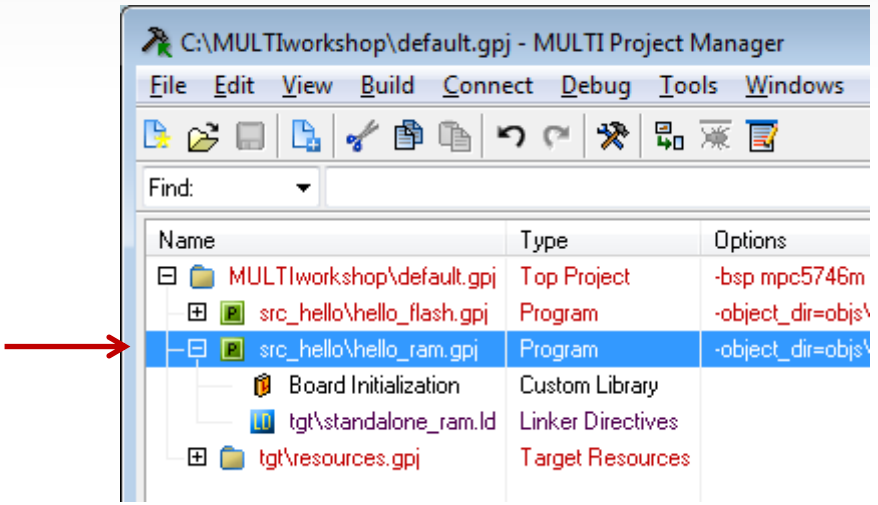
Now the new Program shows up in the Top Project





Hands-on 2.4: **Add Source Files(s)**

- **Right click** on the new **Program** & select "**Add File into hello_ram.gpj**"

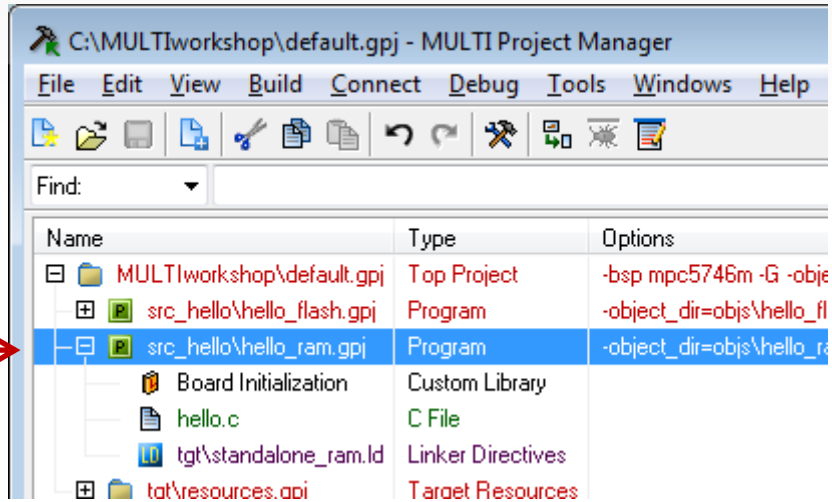


- Select **hello.c** file
- Click **Add** button

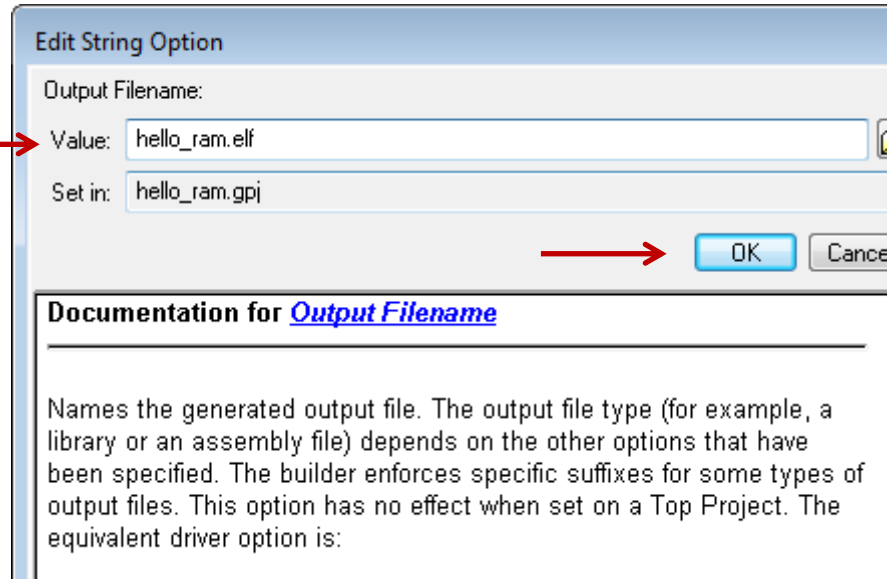
Now the C file shows up in the hello_ram program

Hands-on 2.5: **Add .elf to Output Filename**

- **Right click** on **hello_ram.gpj** and select "**Set Build Options**…"

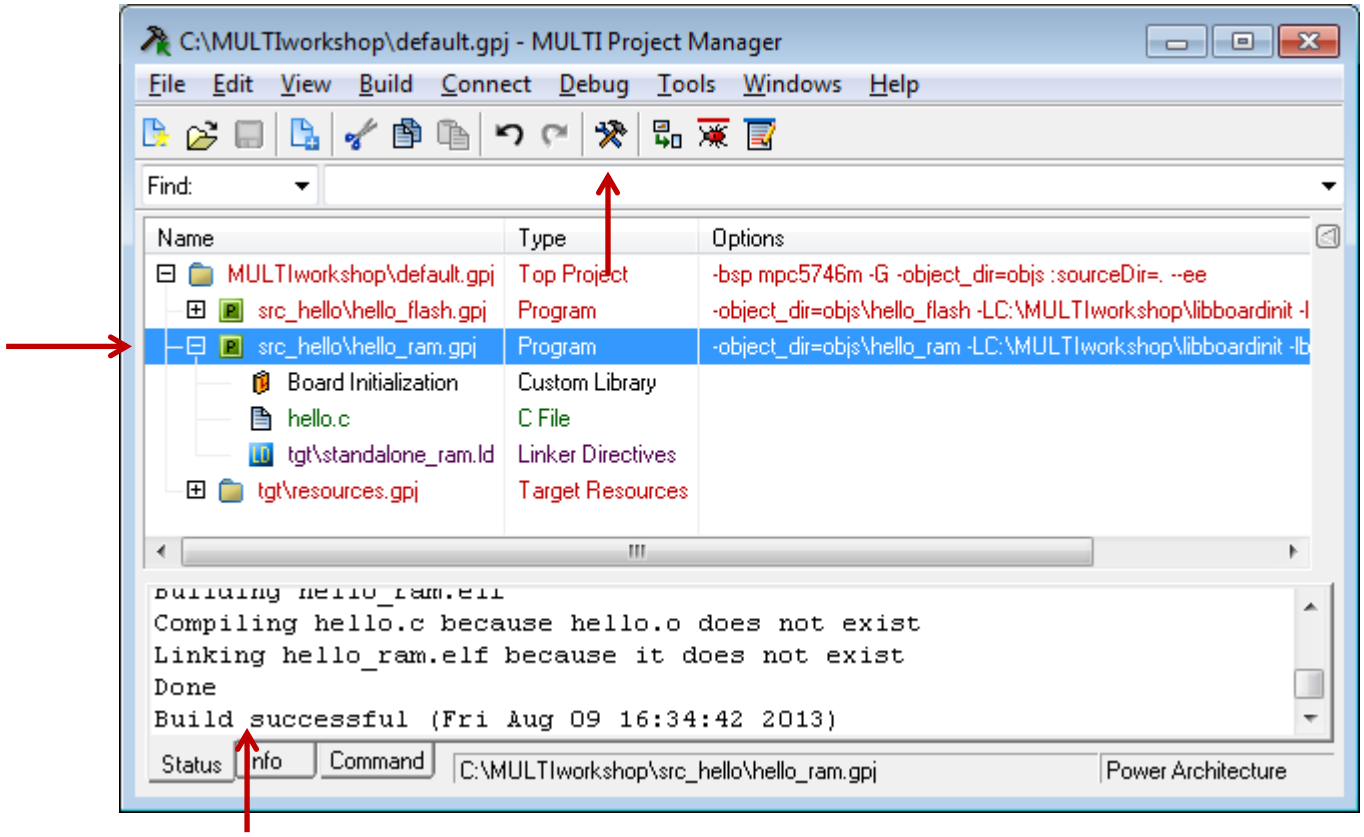


- Select **Project** from the All Options tab
- **Right click** on **Output Filename**
- Click on **Set Output Filename**
- Enter Output File name: **hello_ram.elf**
- Click **OK**
- Click **Done**



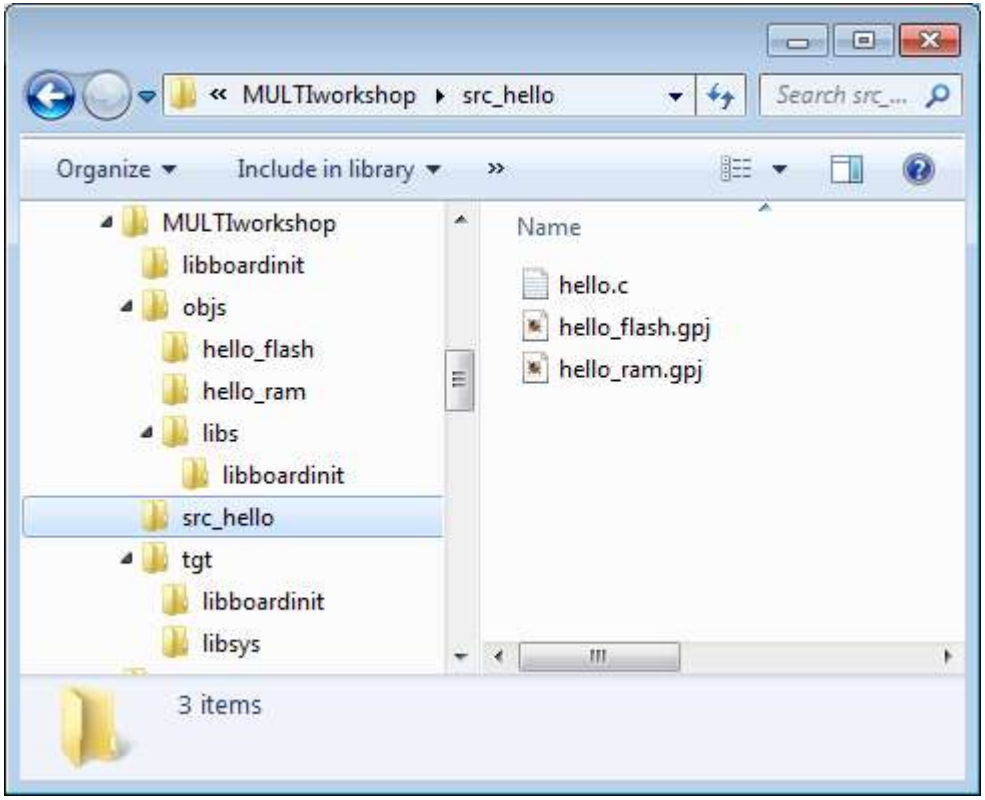
Hands-on 2.6: **Build Project**

- Click on the crossed tools to Build
- “Build successful” should appear in status window



Hands-on 2: **Directory Structure**

The built project has folders and files as shown below



Hands-on 2: **Summary – Add Program**

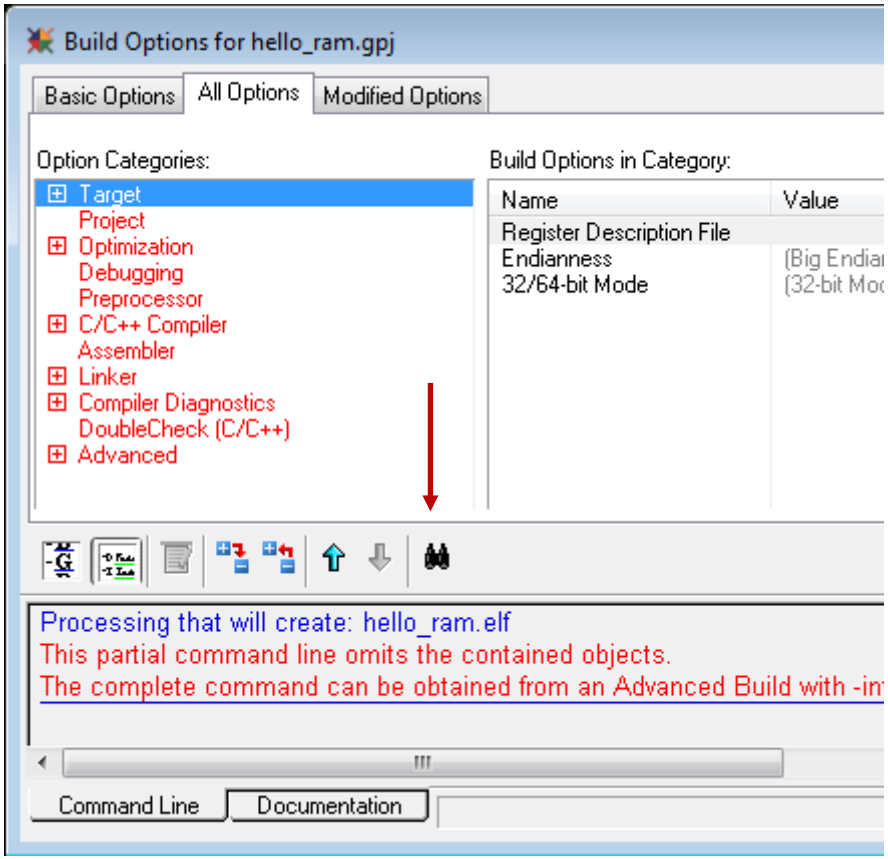
Step			hello_ram target
Add Program – minimal RAM (Project Manager)	1	Add item to top project	Select Program
	2	• Specify source directory • Specify program name	• C:\MULTIworkshop\hello_src • hello_ram
	3	• Configure linker • Select libraries	• Link & execute from ram • (none for ram target)
	4	Add desired source files	hello.c
	5	Debugger compatibility	Add .elf to output file name
	6	Build program	Build

Hands-on 2: **MULTI Tip – Binocular - 1**

Use Binocular tool to locate options in the Builder GUI

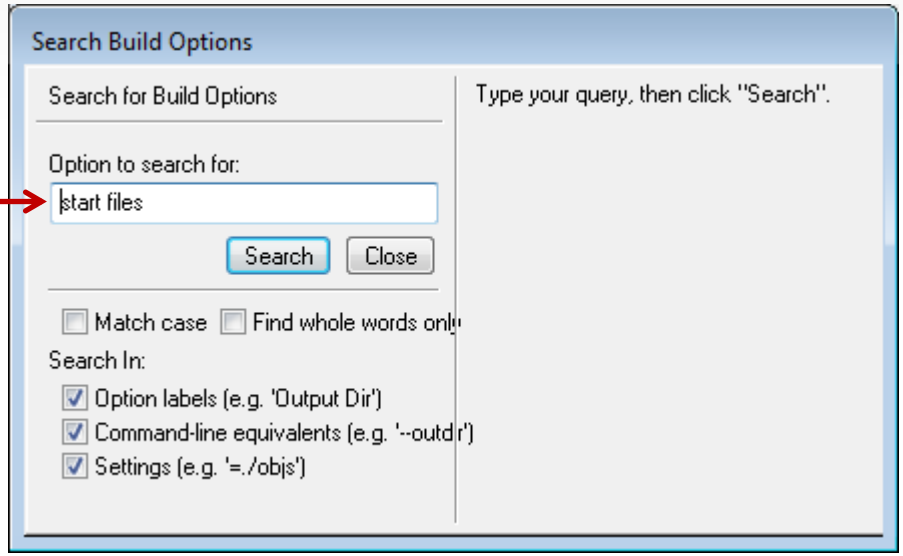
- In a program's "Set Build Options…" window, click on the binoculars icon

Sometimes you want to use your own startup file(s) which requires altering default start file build options. The Binocular tool will be used to find where they are located.

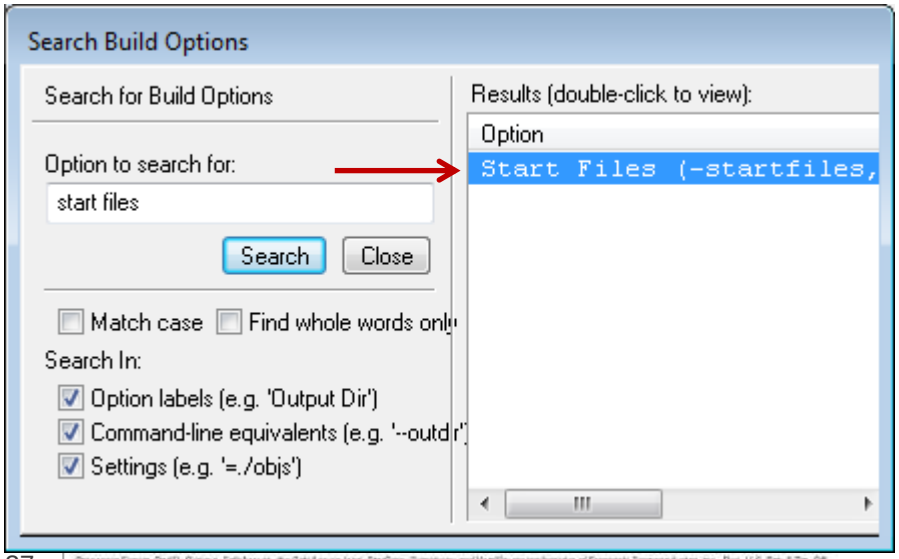


Hands-on 2: **MULTI Tip – Binocular - 2**

- Enter “start” or “start files” & click on Search button



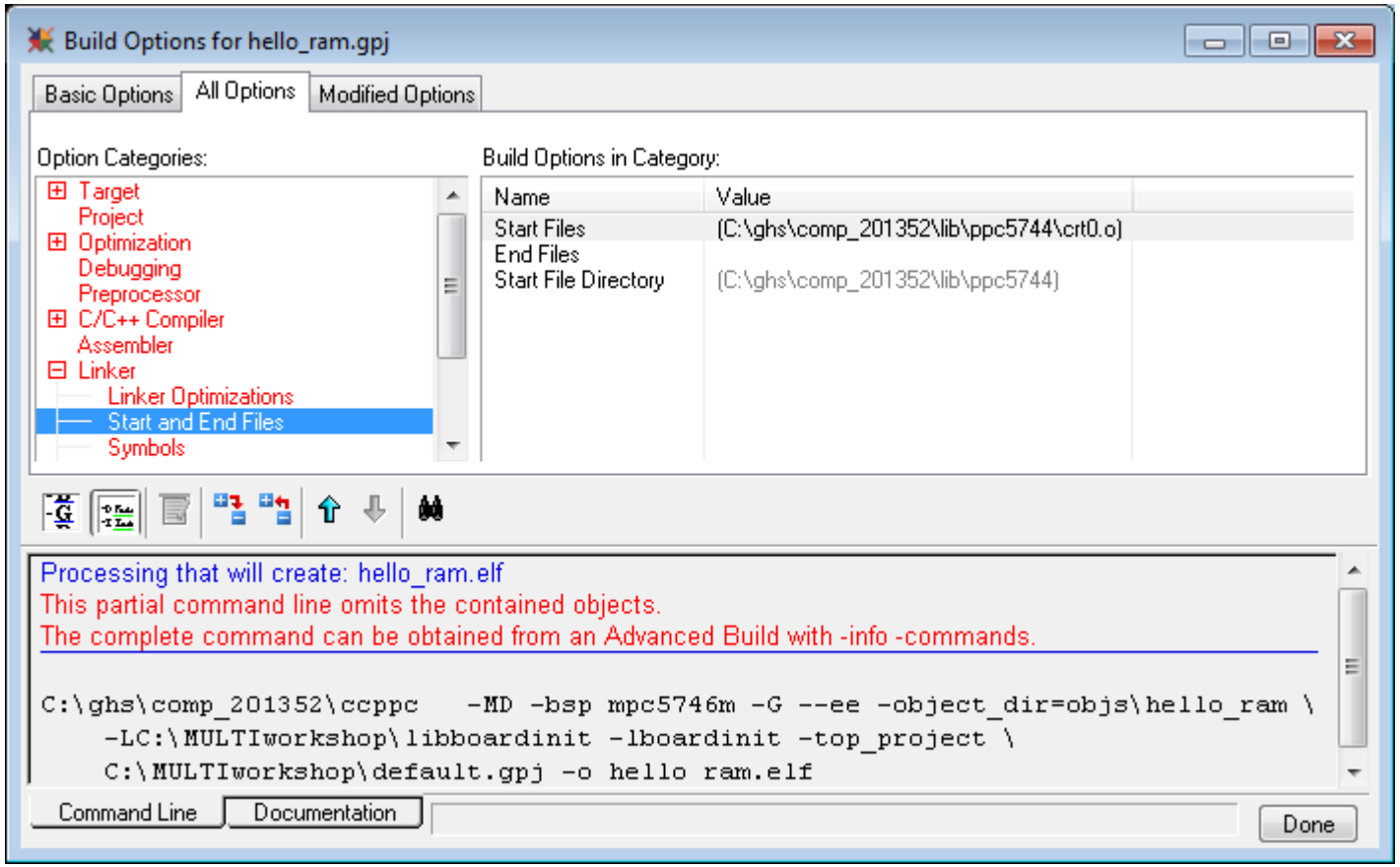
- The Search Build Options window appears.
- Double click on desired item as shown for search of “start”



freescale™

Hands-on 2: **MULTI Tip – Binocular - 3**

- The result location for Start Files is shown along with Documentation at the bottom of the window



Hands-on 3:

- Create a program in the same project using external source code. This program will then be used in Hands-on 4 using the debugger.

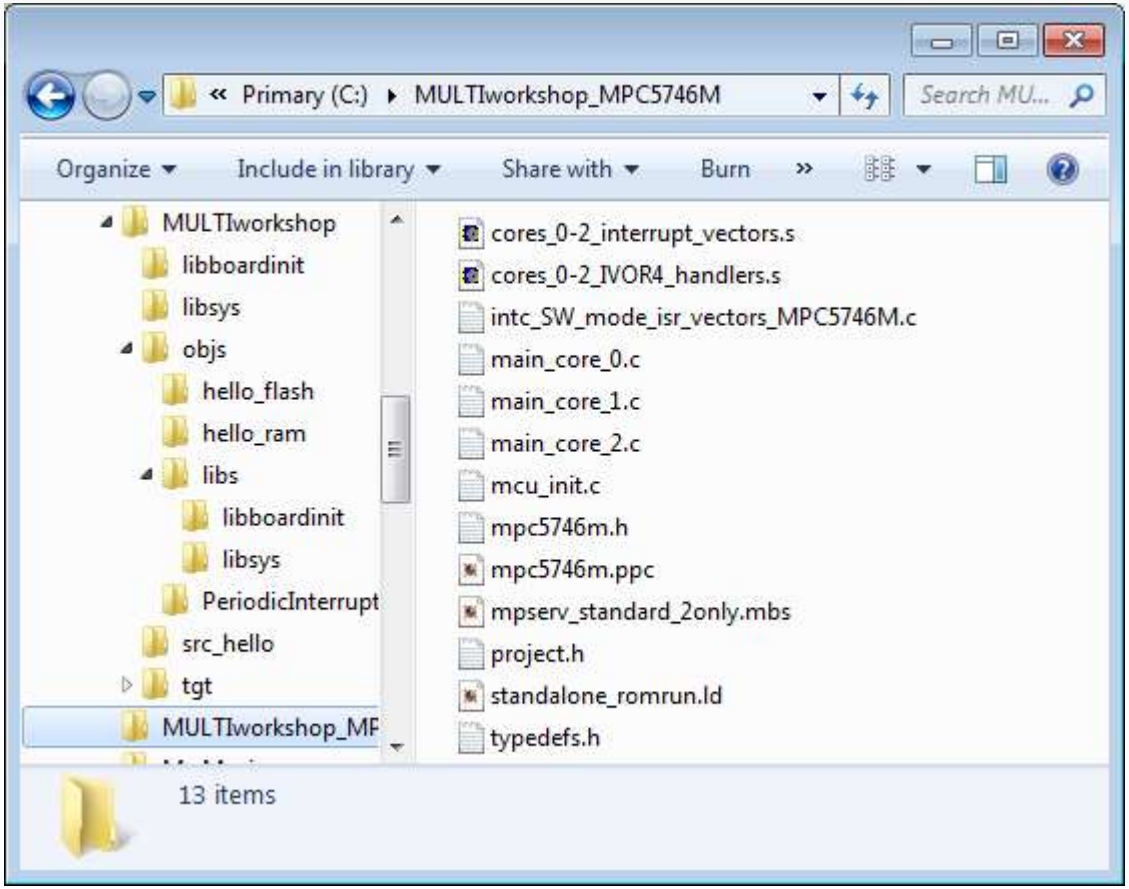
Hands-on 3: **Import Existing Files to Project**

- Files for a task scheduler program will be added to the existing MULTI workshop project
- Interrupts are used to toggle LEDs on the Freescale evaluation board per the table below.

LED	ISR Controlling LED	Core Executing	Toggle Period
LED 0	PIT 0 Timer 0	Core 0	0.5 sec (pit 0 timer 0 timeout)
LED 1	PIT 0 Timer 1	Core 1	1 sec (pit 0 timer 1 timeout)
LED 2	PIT 0 Timer 2	Core 2	2 sec (2000 x pit 0 timer 2 timeouts)
LED 3	Software IRQ #4	IRQ: Core 0 ISR: Core 1	4 sec (8 Core 0 pit IRQs x 0.5 sec/IRQ)

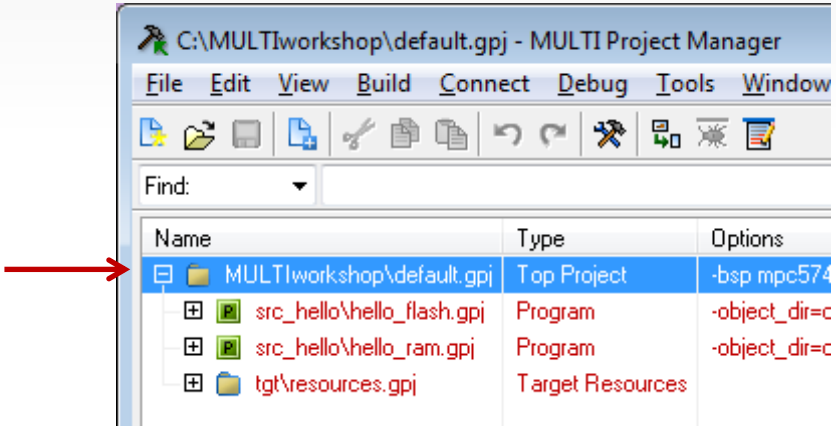
Hands-on 3: **Task Scheduler Application Files**

- **main_core_[x].c**
  - General initialization
  - Configures PIT IRQ
  - Has PIT ISR
- **mcu_init.c**
  - Configures PLL, clocks
  - Activates PLL, clocks
- **cores_0-2_interrupt_vectors.s**
  - Configures core interrupt table (“ivors”)
- **IVOR4_handlers.s**
  - Saves registers on stack
  - Determines ISR vector
  - Branches to ISR
  - Restore registers after ISR
- **intc_SW_mode_isr_vectors_MPC5746M.c**:
  - ISR vector table for MPC5746M INTC in software vector mode

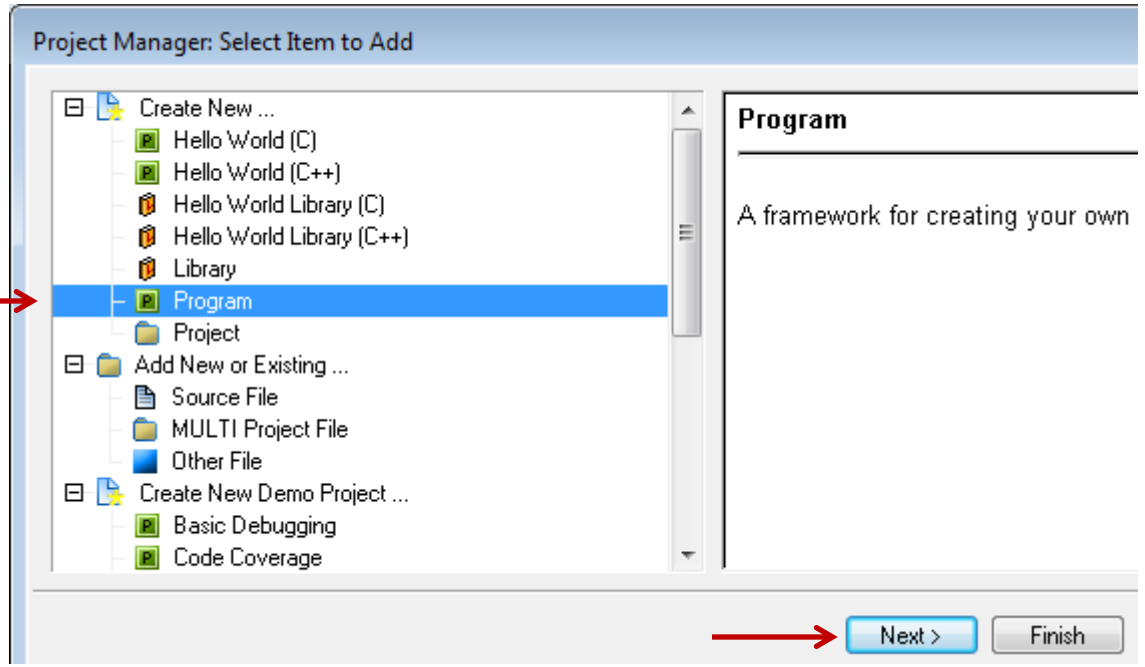


Hands-on 3.1: **Add Program to Top Project**

- **Right click** on **Top Project** & select "**Add Item into default.gpj**…"

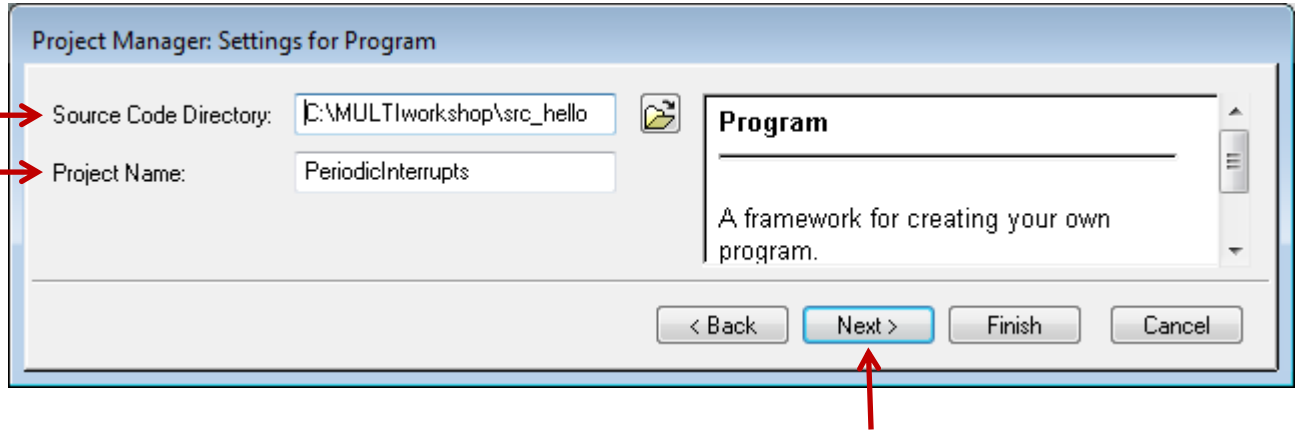


- Select **Program** in the Create New options
- Click **Next**



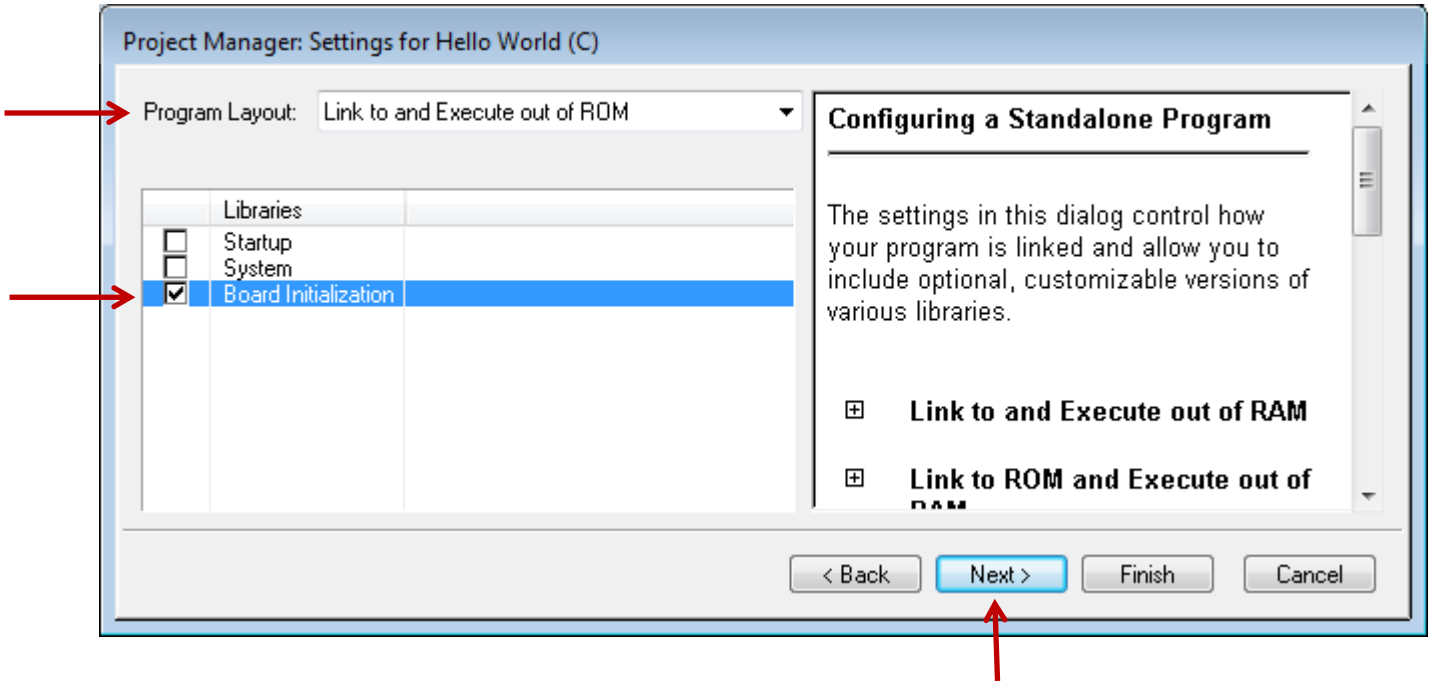
Hands-on 3.2: **Specify Source and Name**

- **Source Code Directory**: change to use same as before:
  C:\MULTIworkshop\src_hello
  - type it in so project will use a relative path
- Project Name:
  - rename to **PeriodicInterrupts**
- Click **Next**



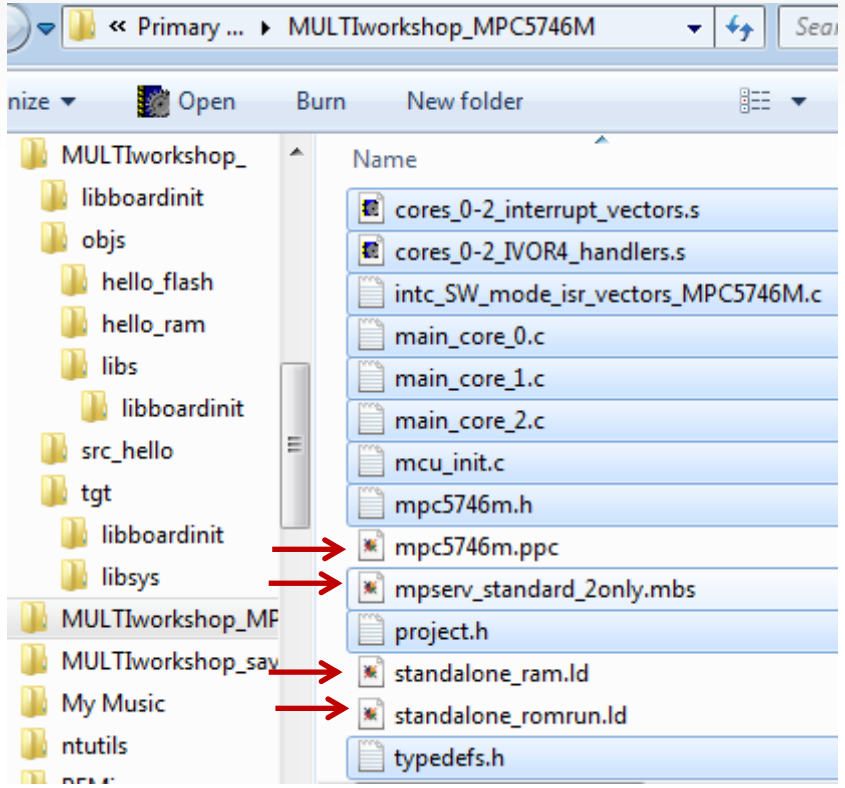
Hands-on 3.3: **Configure Linker and Libraries**

- **Program Layout**: Select "**Link to and Execute out of ROM**"
- **Libraries**: Select "**Board Initialization**" to run from flash
- Click **Next**

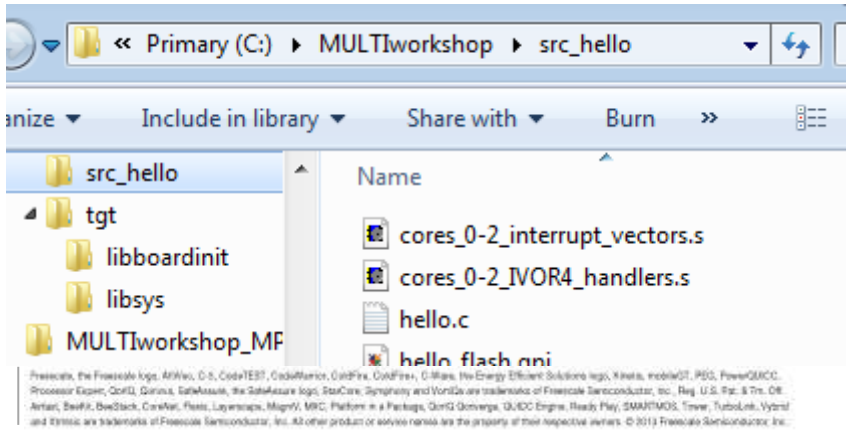


Hands-on 3.4: **Add files 1: Copy Source Files**

- **Navigate** in Windows Explorer to
  c:\MULTIworkshop_MPC5746M
- **Select** only the files shown.
  - Do not select .ppc, .mbs, or .ld files
- **Copy** files to the clipboard (Ctrl-C)

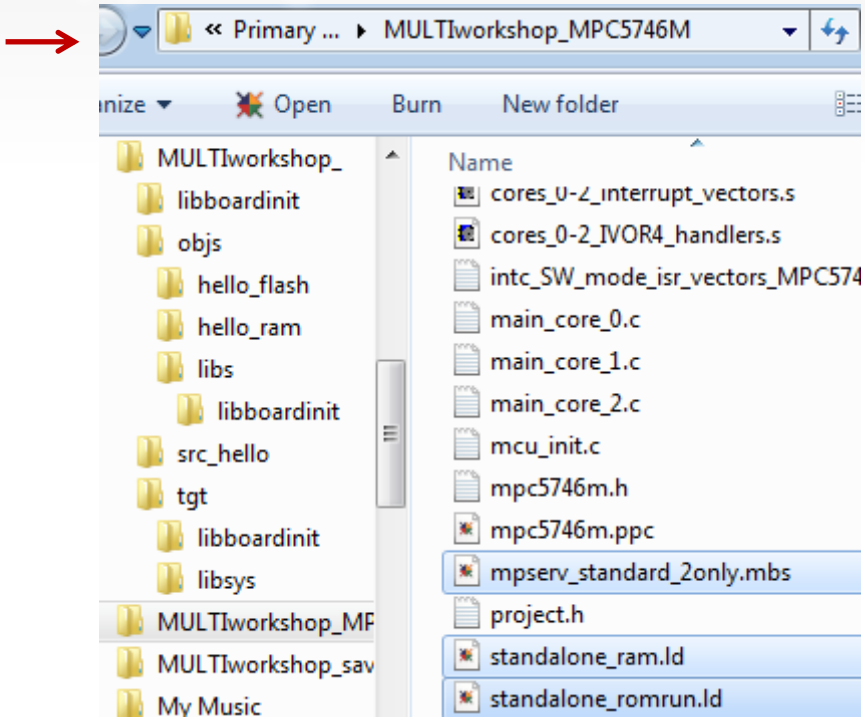


- **Navigate** to the project's source directory
  c:\MULTIworkshop\src_hello
- **Paste** files

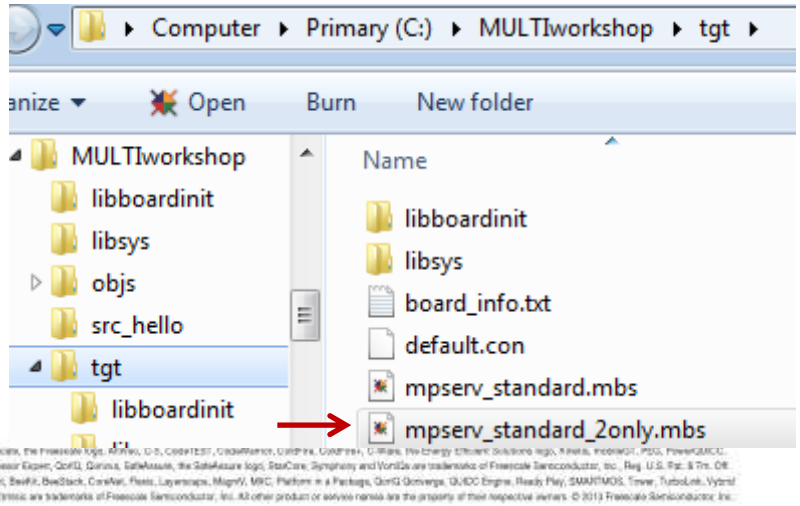


Hands-on 3.4: **Add files 2: Copy Target Files**

- **Navigate** in Windows Explorer to
  c:\MULTIworkshop_MPC5746M
- **Select** the files**:**
  - **mpserve_standard_2only.mps**
  - **Standalone_ram.ld**
  - **standalone_romrun.ld**
- **Copy** to clipboard

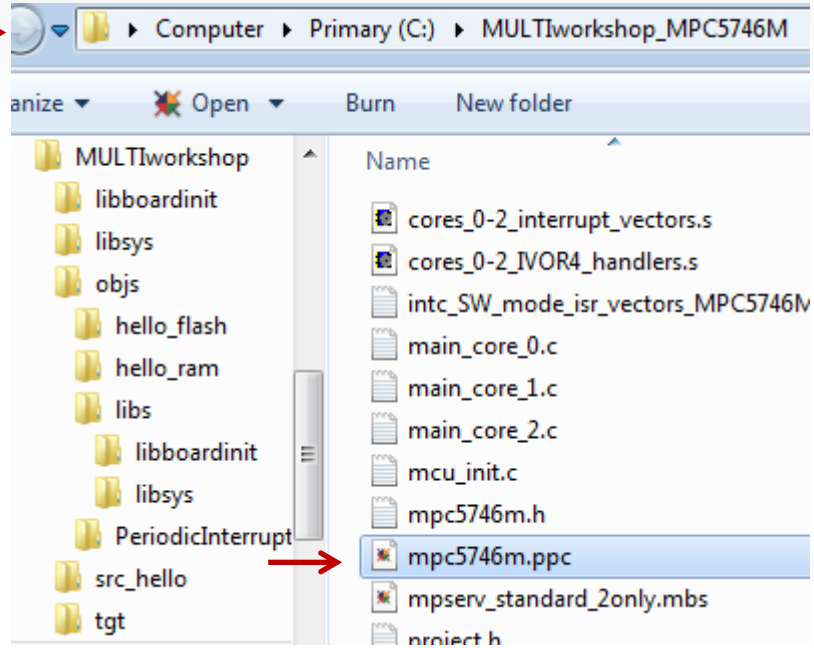


- **Navigate** to the project's tgt directory
  c:\MULTIworkshop\tgt
- **Paste** the files (replace if asked)
  - Adds mpserve_standard_2only.mps
  - Replaces standalone_romrun.ld
    - Newer version adds interrupt sections

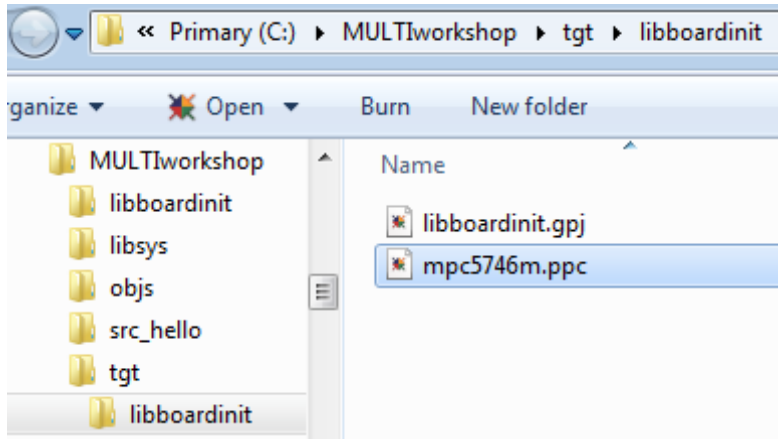


Hands-on 3.4: **Add files 3: Copy New Chip Init. File**

- **Navigate** in Windows Explorer to
  c:\MULTIworkshop_MPC5746M
- **Select** the file**:**
  - **mpc5746m.ppc**
- **Copy** to clipboard

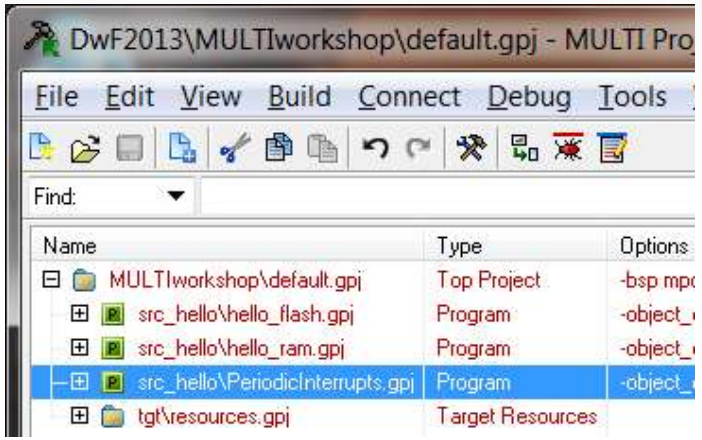


- **Navigate** to the project's tgt/libboardinit directory
  c:\MULTIworkshop\tgt\libboardinit
- **Paste** the file (replace file)
  - Updated file has initialization "tweaks"

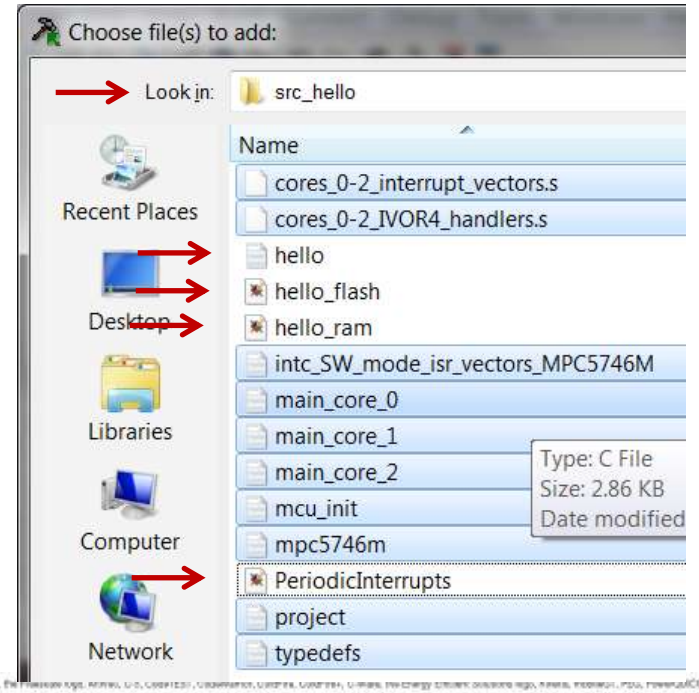


Hands-on 3.4: **Add Files 4: Import Files to Project**

- **Right click** on the Program and select "**Add File into..**"

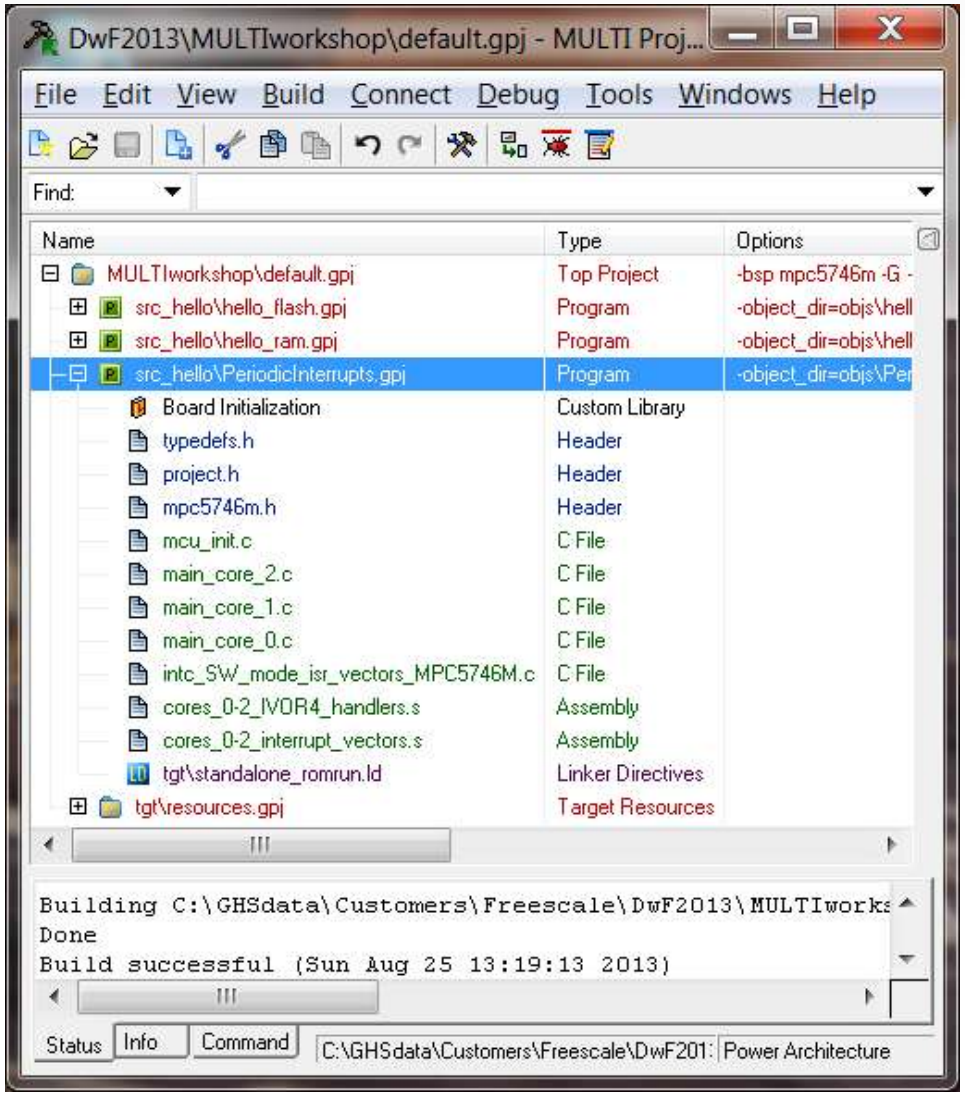


- **Navigate** to the project source folder, src_hello
- **Select only the files shown.** Do not select the .gpj files or hello.c.
  - If necessary to see all files, change "Files of type" to "All files".
- Click **Add**



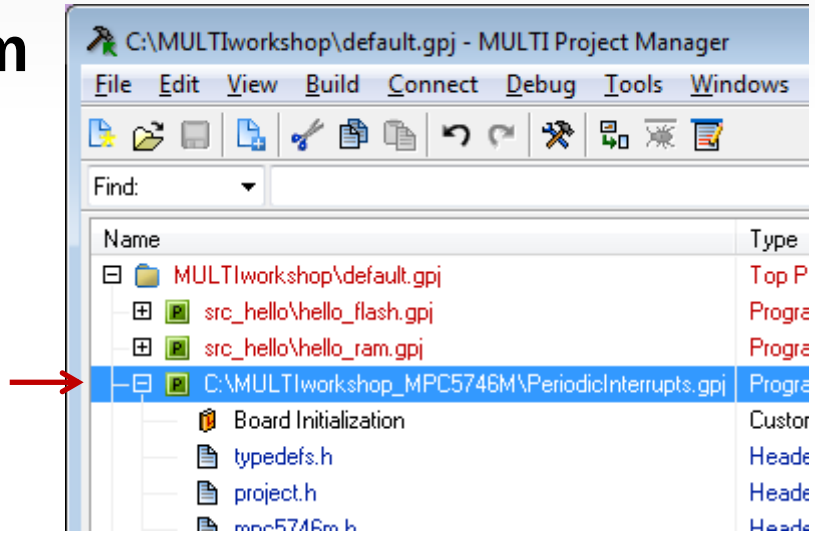
Hands-on 3.4: **Add Files 5: Completed Project**

- Added files are now shown in subproject

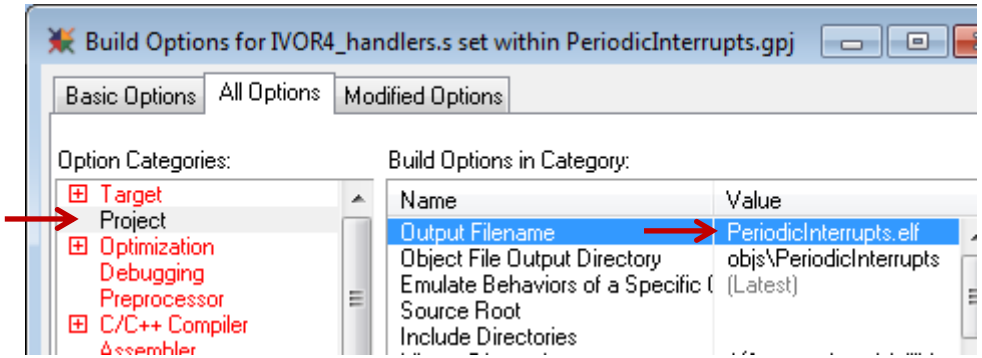


Hands-on 3.5: **Add .elf to Output Filename**

- **Right click** on **Program**
- Select "**Set Build Options**"

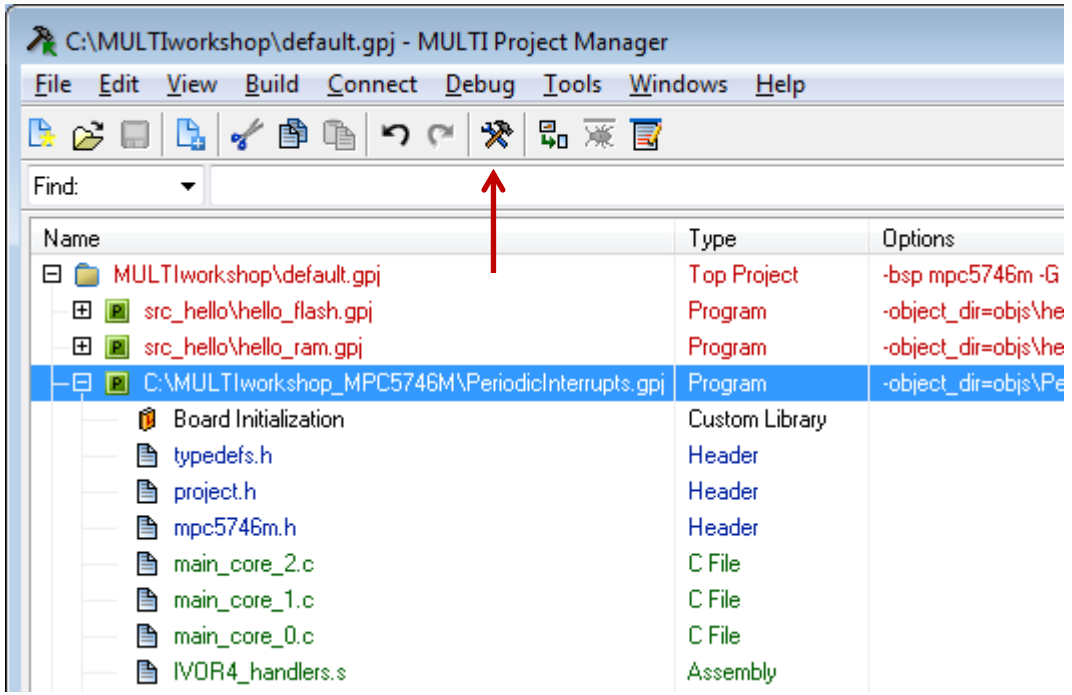


- In All Options tab, select **Project**
- **Double Click** on "**Output Filename**"
- Output Filename: change to **PeriodicInterrupts.elf**
- Click **OK**
- Click **Done**



Hands-on 3.6: **Build Program**

- Select the Program
- Click on build tool



Hands-on 3: **Summary – External File Program**

Step			PeriodicInterrupt target
Add Program with External Files (Project Manager)	1	Add item to top project	Add Program
	2	• Specify source directory • Specify program name	• C:\MULTIworkshop\hello_src • TaskScheduler
	3	• Configure linker • Select libraries	• Link & execute from ROM • Board init
	4	Add files to project	Add from separate folder: • Source files • Modified link file • Modified debugger script file • Modified chip initialization file
	5	Debugger compatibility	Add .elf to output filename
	6	Build program	Build

Hands-on 3: **MPC5746M Boot Header**

Section	.long value	Description
.boot _header	0x005A 0001	Boot header id + enable CPU2
	__ghs_rombootcodestart	CPU2 reset vector
	0x0	(reserved)
	0x0	(reserved)
	0x0	CPU0 reset vector
	0x0	CPU1 reset vector
	0x0	CPUC reset vector
	0x0	(reserved)

Notes:

- CPU2 is the only active core when connecting with a debugger. By default, the BAF will not run from the debugger other than on CPU2.
- Boot header for GHS project is in file mpc5746m.ppc

Start Up Flow - 1 (MPC5746M project)

	mpc5746m.ppc	crt0.ppc
reset	.boot_header at 0xF8_C000 (16KB flash block 0) • one CPU (CPU2) is enabled for initial boot • boot core vector: __ghs_rombootcodestart = start of .text	
1		_start (1st line code section): • bl __ghs_board_memory_init
2	__ghs_board_memory_init: • ECC init shared sram (fills with branch to self instructions) • bl __ghs_e200z4204_core_init	
3	__ghs_e200z4204_core_init: • MSR[ME] = 1 • IVPR= 0x4000 0000 (will be changed in main) • init local data memory • init i & d caches if they exist	
4		• ghs housekeeping: enable debugger syscals (e.g. printf) • init sp, sda, sda2 to shared memory • ghs housekeeping if PIC • jump __ghs_ind_crt0

Start Up Flow - 2 (MPC5746M project)

	mpc5746m.ppc	ind_crt0.c	ind_crt1.c
5		__ghs_ind_crt0: • intializes static and global data • if PIC, PIR, PID it relocates pointers • assigns pointer and calls __ghs_board_cache_sync	
6	__ghs_board_cache_sync • flushes data cache • invalidates instruction cache		
7		• optional decompression of initialized data • optional checksum • assigns pointer and calls __ghs_ind_crt1	
8			__ghs_ind_crt1: • init run time libraries • exits to main (of boot core)

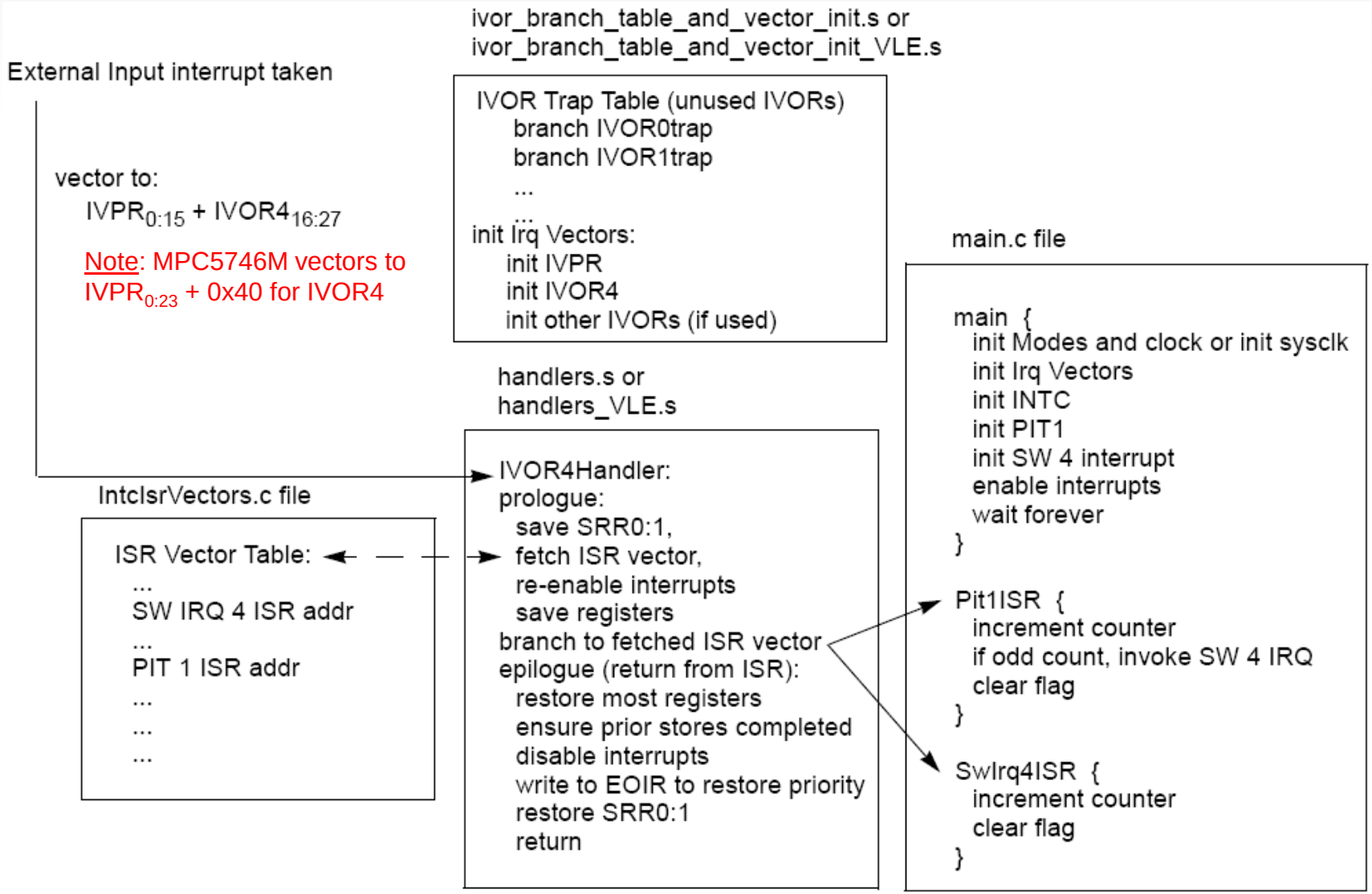
Start Up Flow - 3 (MPC5746M project)

	mpc5746m.ppc	main_core_2.c (boot core)
9		main: • initializations before starting other cores (clocks, etc.) • boot_core_0: If CPU0 is not running per ME_CS, then: 1. Init ME core 0 ctrl reg. (selected: core runs in all RUN modes) 2. Init ME core 0 addr reg. = __ghs_mpc5746m_cpu0_entry 3. Do mode transition to enable core0 to start running
10	__ghs_mpc5746m_cpu0_entry: • init unique sp but common sda & sda2 • if main_core_1 exists, bl to main_core_0 (If there is no main_core_0 then core spins harmlessly)	• boot_core_1: same as above but for core_1

Start Up Flow - 4 (MPC5746M project)

	mpc5746m.ppc	main_core_0.c (boot core)	main_core_1.c	main_core_2.c
11	__ghs_mpc5746m_cpu1_entry: • init unique sp but common sda & sda2 • if main_core_1 exists, bl to main_core_1 (If there is no main_core_1 then core spins harmlessly)	main_core_0		
12			main_core_1	• other functions

Hands-on 3: **Interrupt Program Flow** (ref. AN2865)



Hands-on 4:

- Basic Debugging

Hands-on 4 & 5 Requirements:

- MULTI 6.1.4/Compiler 2013.5.2 and GH probe update 5.0.4.
- Probe target = PPC5746M.
  - To verify, type “target set target” in debugger command window.
  - To change target to PPC5746M, type “target set target PPC5746M”

To configure GHS Probe automaticially for specific target:

- MULTI Launcher -> Utilities -> Probe Administrator
- Select your probe on Probe Administrator list (it may be the only one)
- Probe -> Configure Probe..
- Select the “Prompt” tab and enter “detect”. The device, such as “ppc5746M” will be detected.
- Enter “save” and exit

To Disable Watchdog:

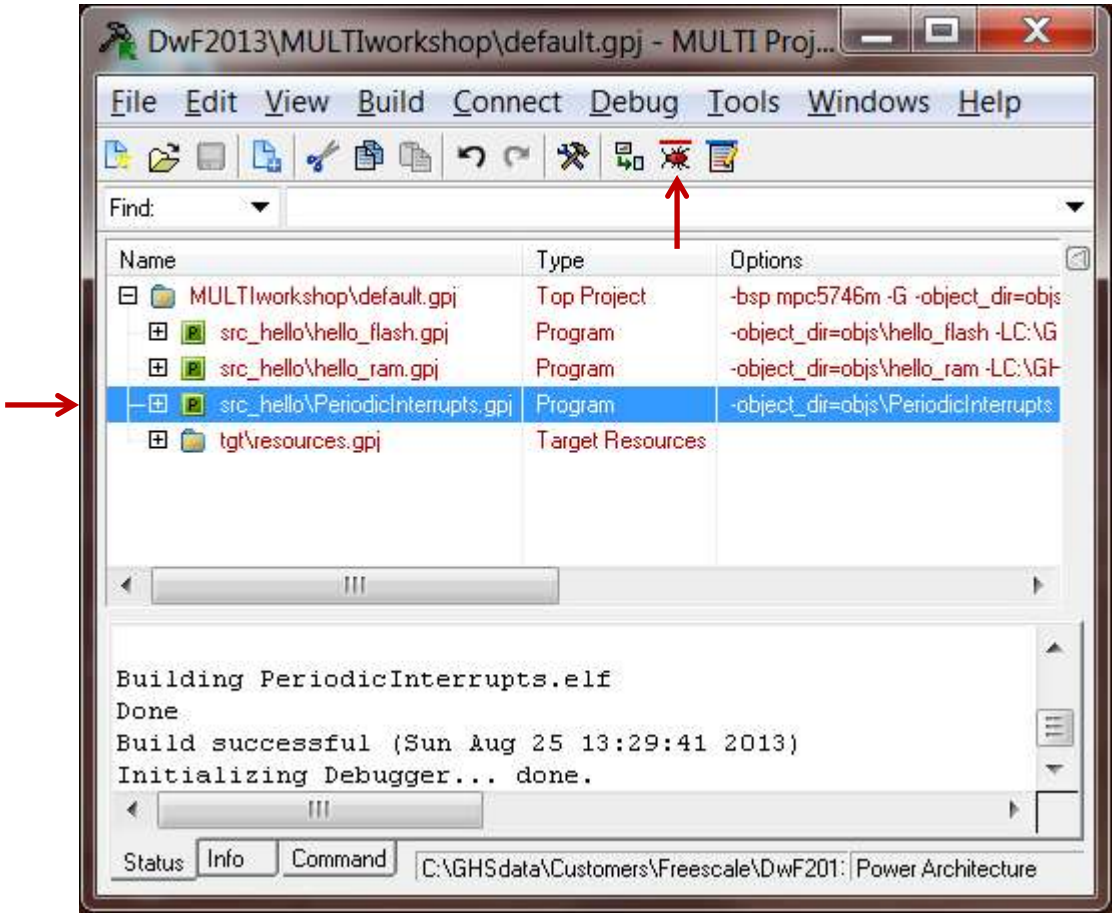
- MULTI Debugger window (after connection) -> “Trg” tab at bottom
- Enter “set” to see options and their syntax. From list, you see: disable_swt
- Enter “disable_swt on”

Hands-on 4: Debug Steps to be Exercised

4.1 – 4.3 Start Debugger, Select Target Connection, Connect to Target
4.4 Program Target Flash
4.5 – 4.6 Navigate to Main, Run to Hardware Breakpoint
4.7 Go to Definition
4.8 – 4.9 Step Over, Step In
4.10 – 4.11 Navigate & View Call Stack
4.12 – 4.13 View & Change a Structured Element
4.14 – 4.15 Display Peripheral Register, Display Structure Definition
4.16 Mixed Mode
4.17 Run, Halt Execution
4.18 Display Memory
4.19 Display Variables while Running
4.20 Using a Complex Breakpoint
4.21 Using a Data Breakpoint
4.22 Multicore Debugging

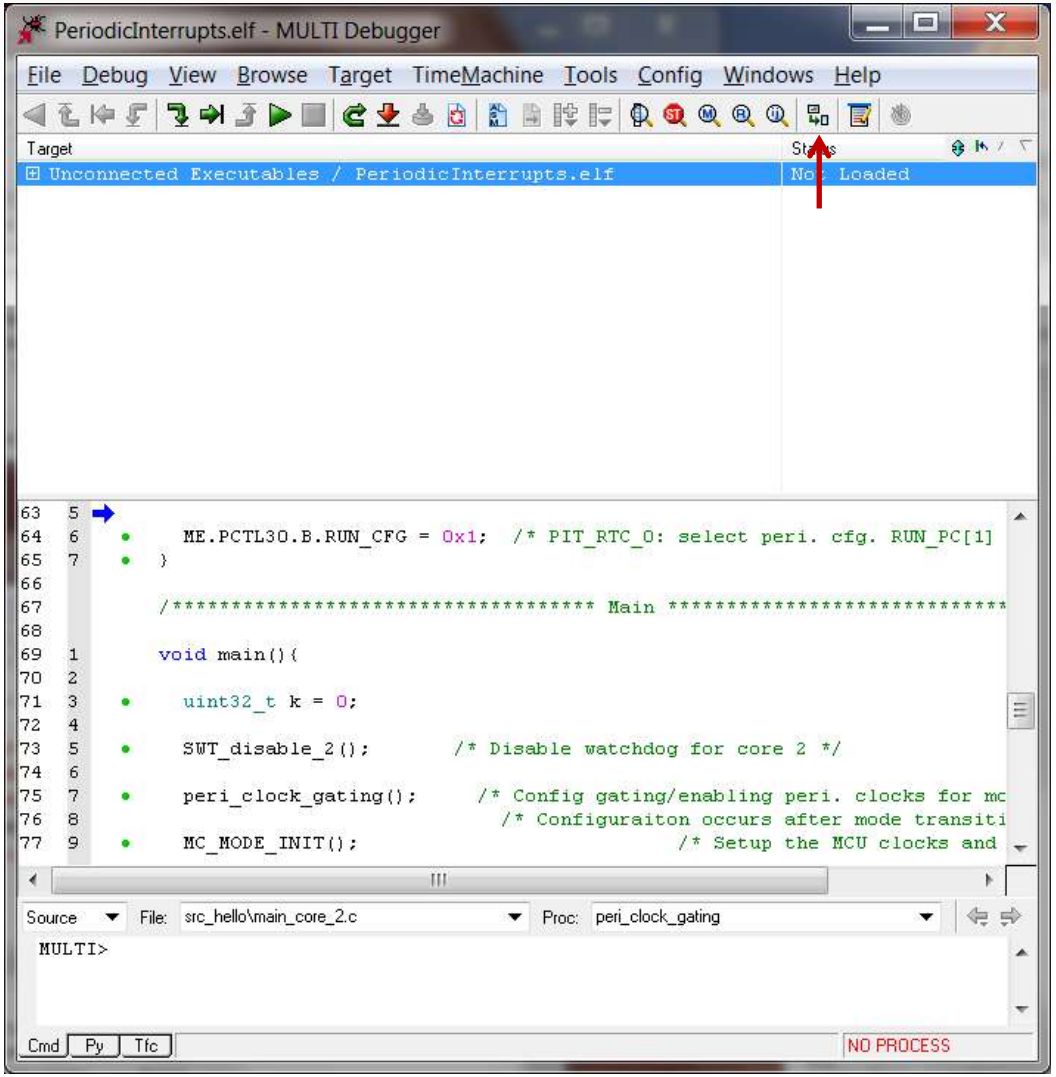
Hands-on 4.1: **Start the Debugger**

- Select the **PeriodicInterrupts program** in the Builder window
- **Left click** on the "**bug**" button to start the Debugger



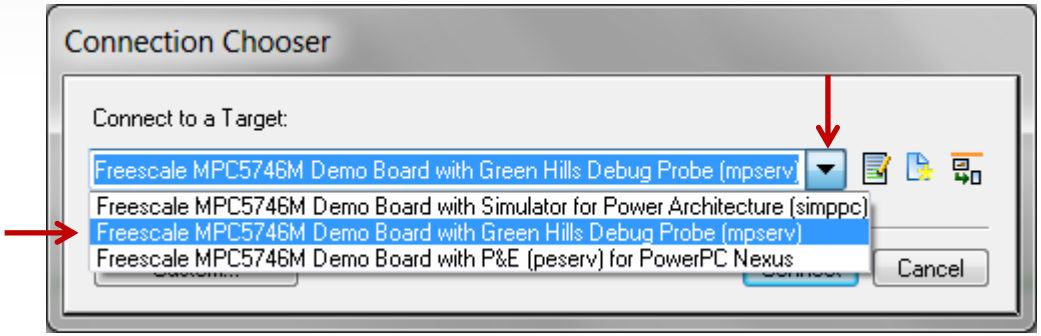
Hands-on 4.2: **Select a Target Connection - 1**

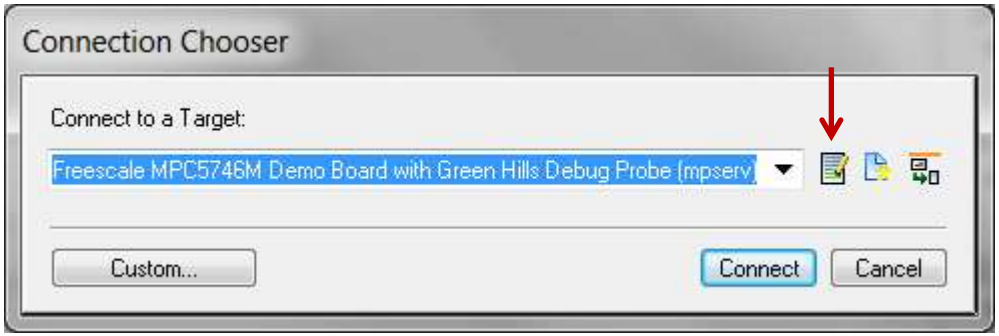
- **Left click** on the "**Connect**" button in the Debugger window



Hands-on 4.2: **Select a Target Connection - 2**

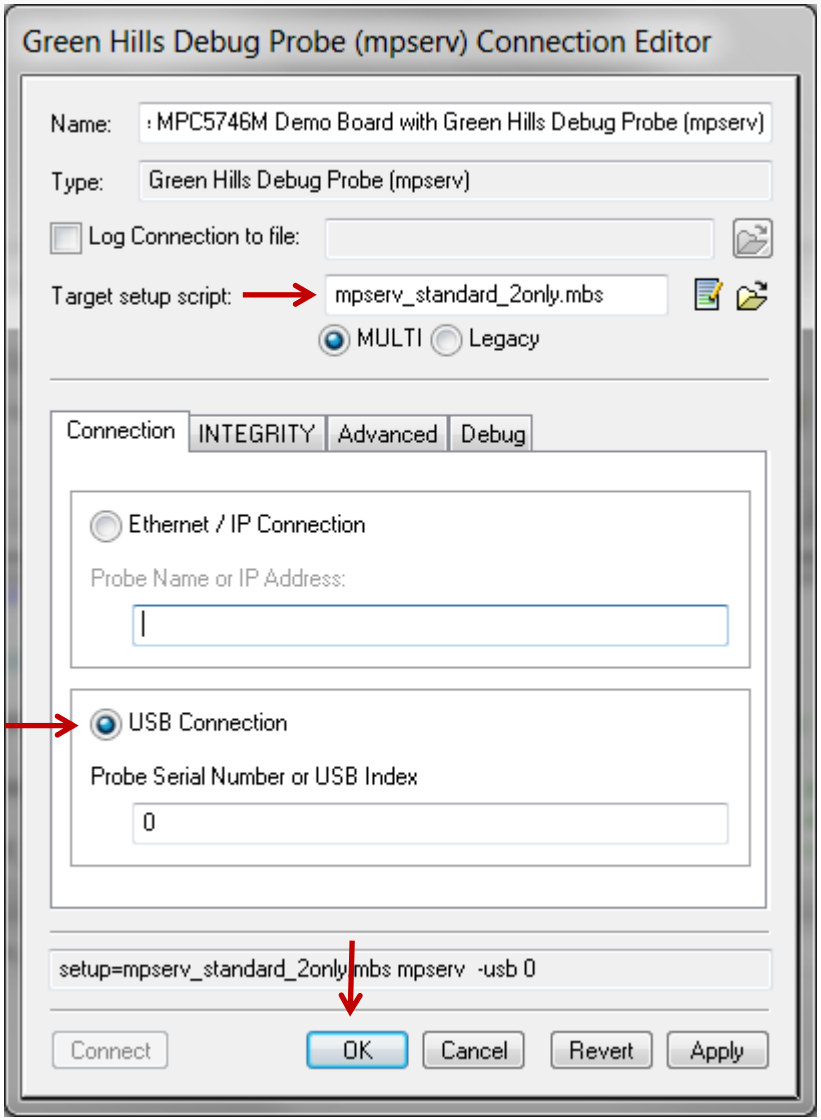
- **Left click** on the "**Connect to a Target**" pull down menu
- **Left click** on the GH Probe entry "**mpserv**"
  - Note there is also a "peserv" entry for PE Micro hardware
- **Left click** on the "**Edit**" button





Hands-on 4.2: **Select a Target Connection - 3**

- **Left click** on **USB Connection**
- Modify the target setup script to use mpserv_standard_2only.mbs
- **Left click** on “OK”

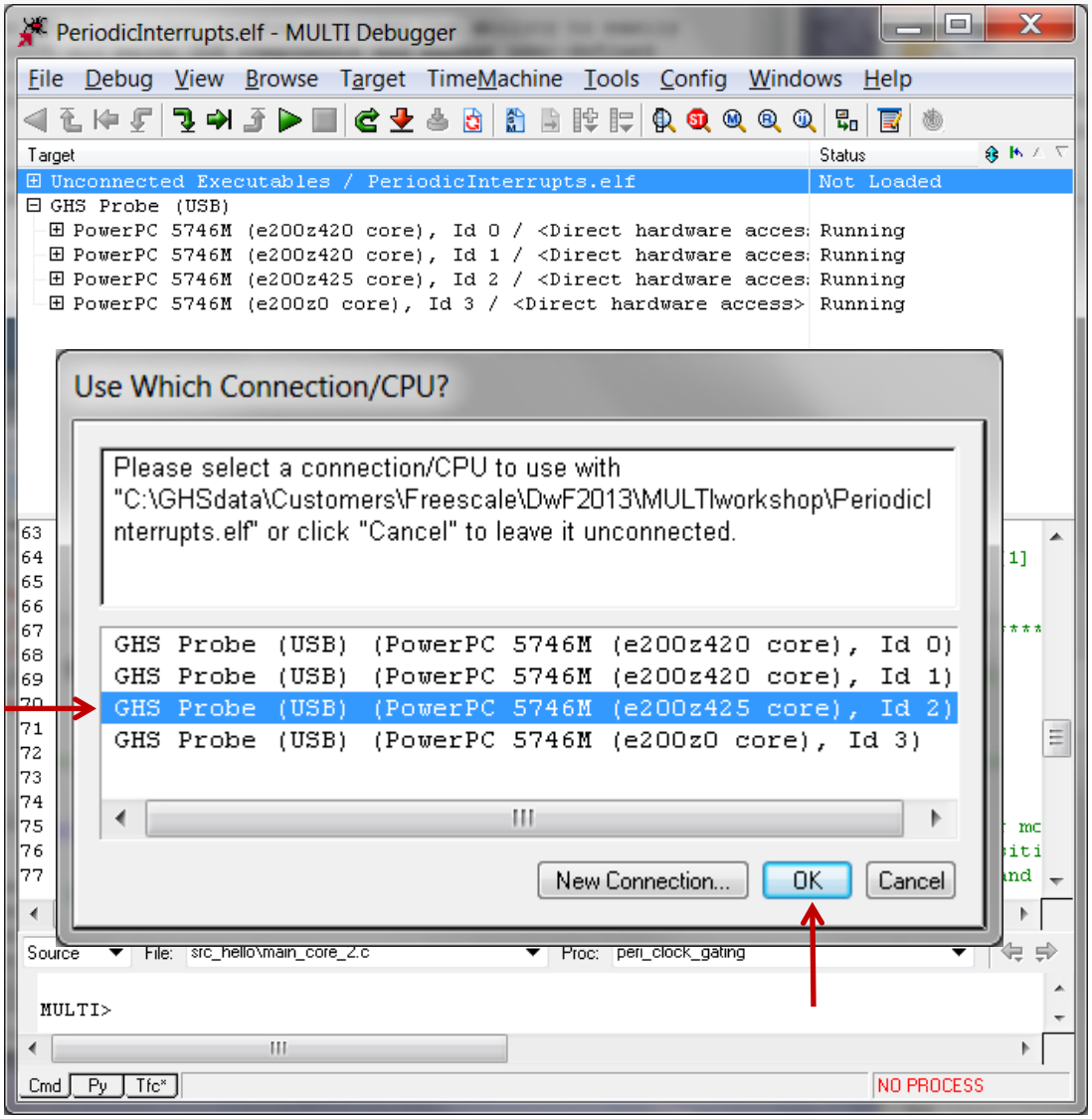


Hands-on 4.3: **Connect to the Target 1**

- Make sure the GH Probe and target are powered on
- **Left click** on the "**Connect**" button

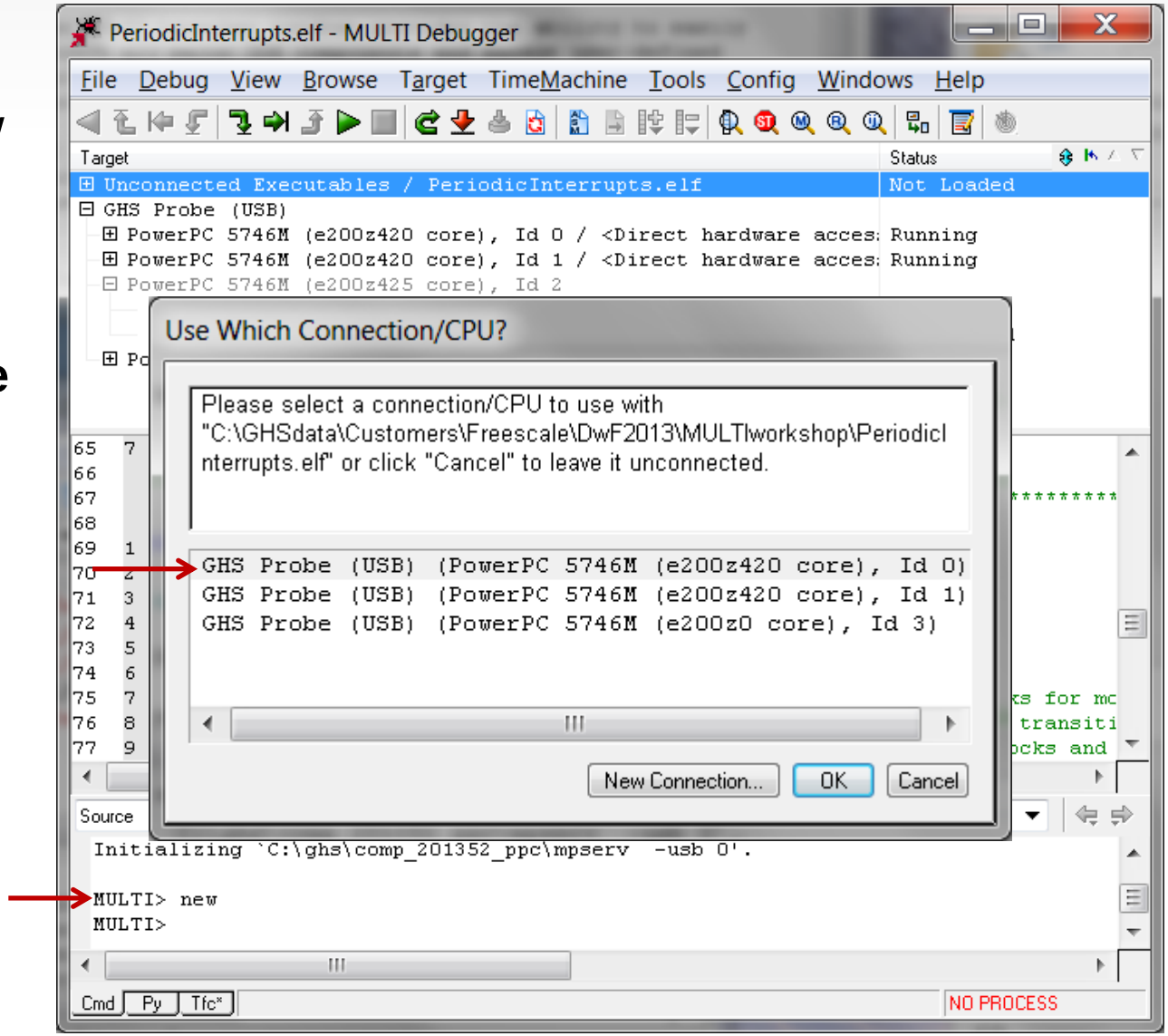
Hands-on 4.3: **Connect to the Target 2**

- The four cores appear in the debugger window and a CPU selection window opens
- **Left click** on **core ID 2** (this core receives hardware reset by default in the MPC5746M)
- **Left Click** on **OK** to continue



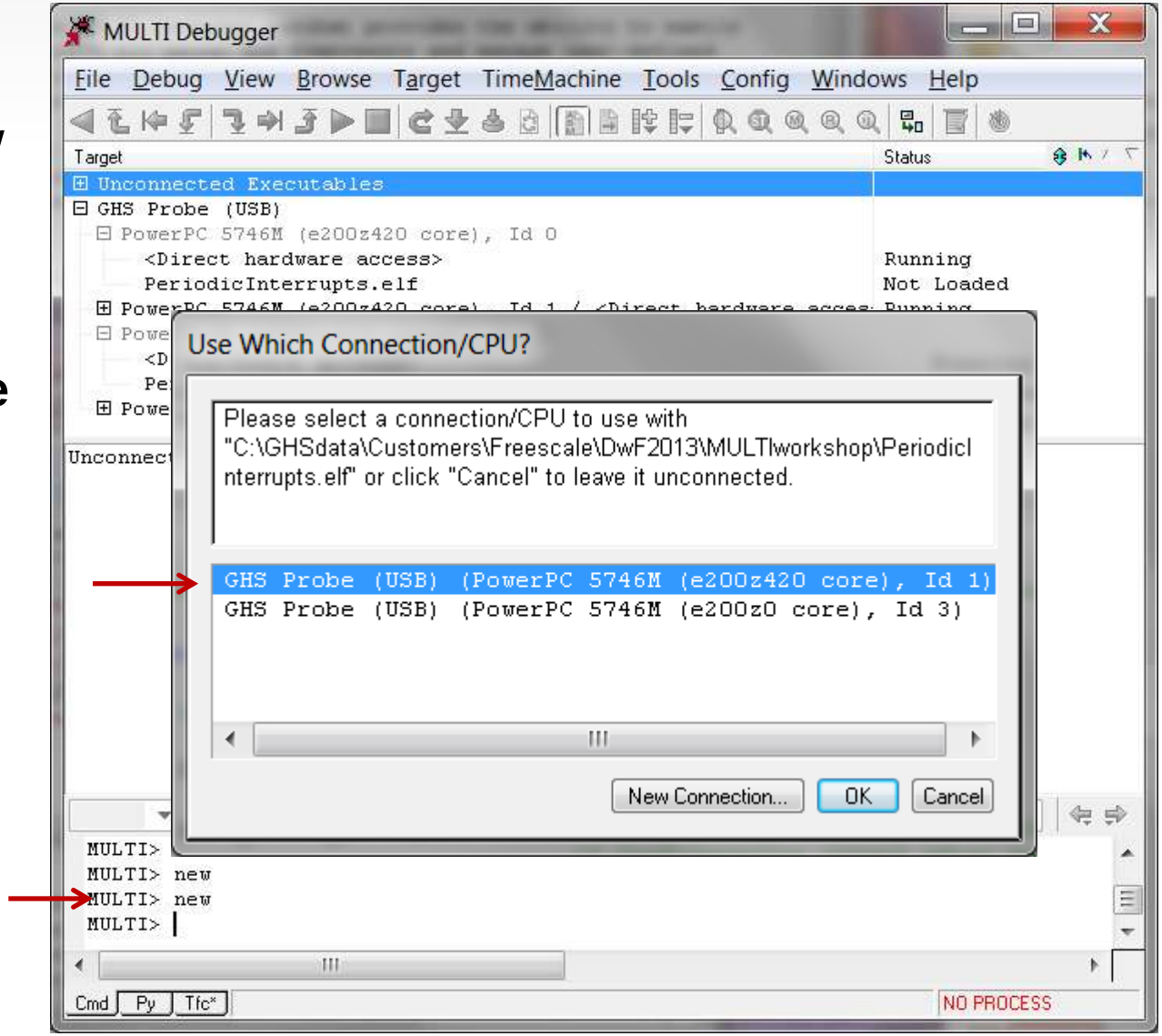
Hands-on 4.3: **Connect to the Target 3**

- Enter “new” in the command window and the CPU selection window opens again
- **Left click** on **core ID 0** for the MPC5746M
- **Left click** on **OK** to continue



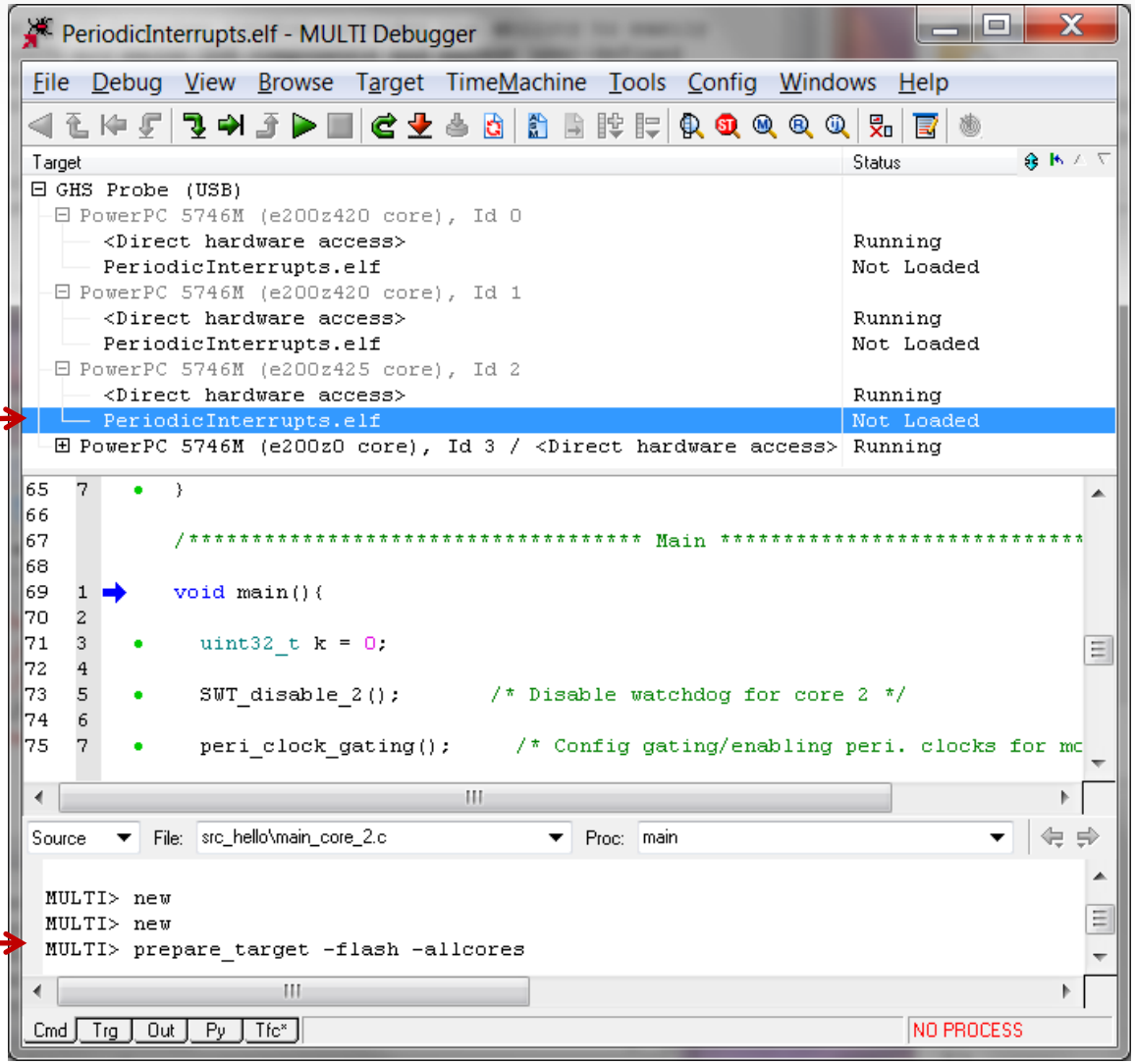
Hands-on 4.3: **Connect to the Target 4**

- Enter “new” in the command window and the CPU selection window opens again
- **Left click** on **core ID 1** for the MPC5746M
- Click on **OK** to continue



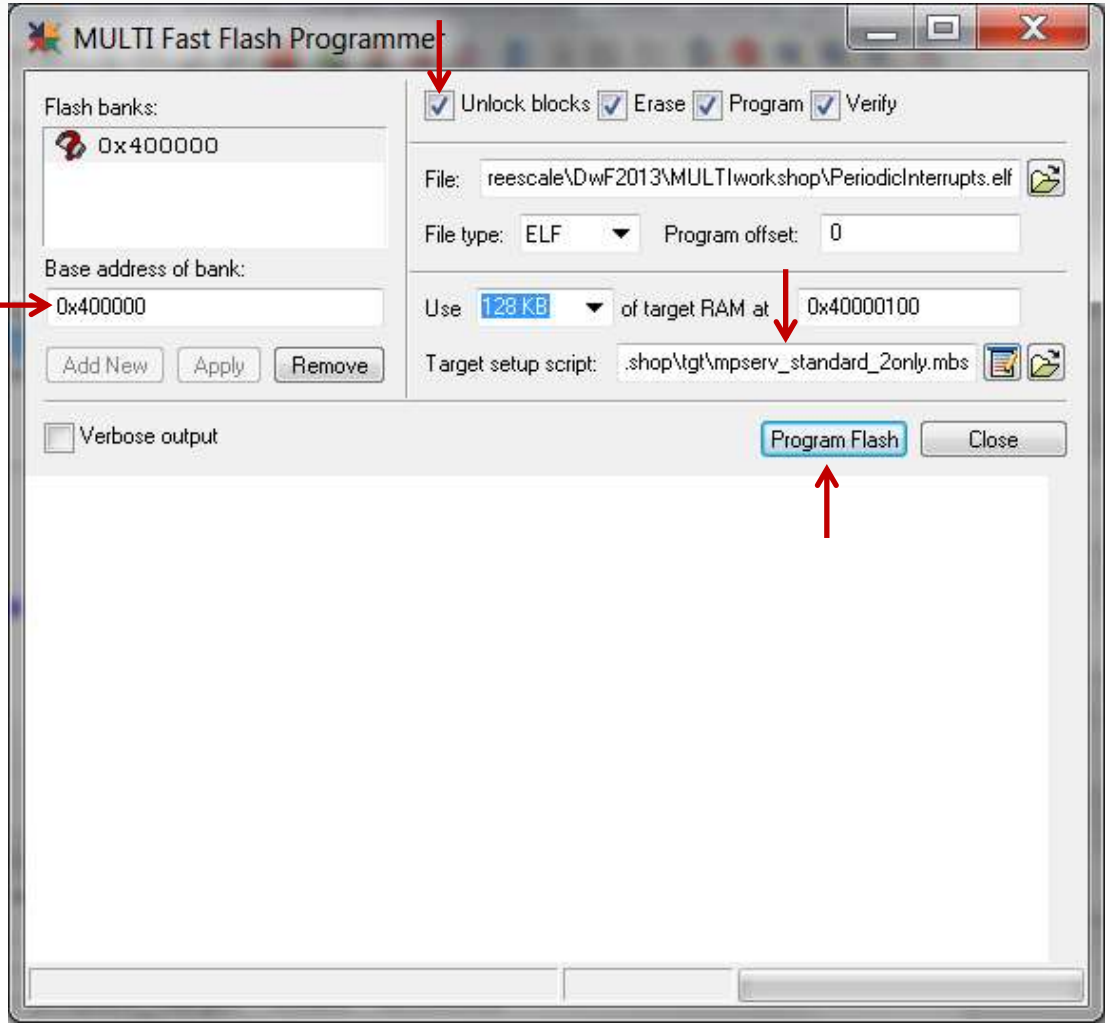
Hands-on 4.4: **Program Target Flash Memory - 1**

- **Left click** on the application PeriodicInterrupts.elf for core ID 2 (skipping this step is a common mistake)
- Enter "**prepare_target –flash –allcores**" in the command window and the flash programmer window opens.



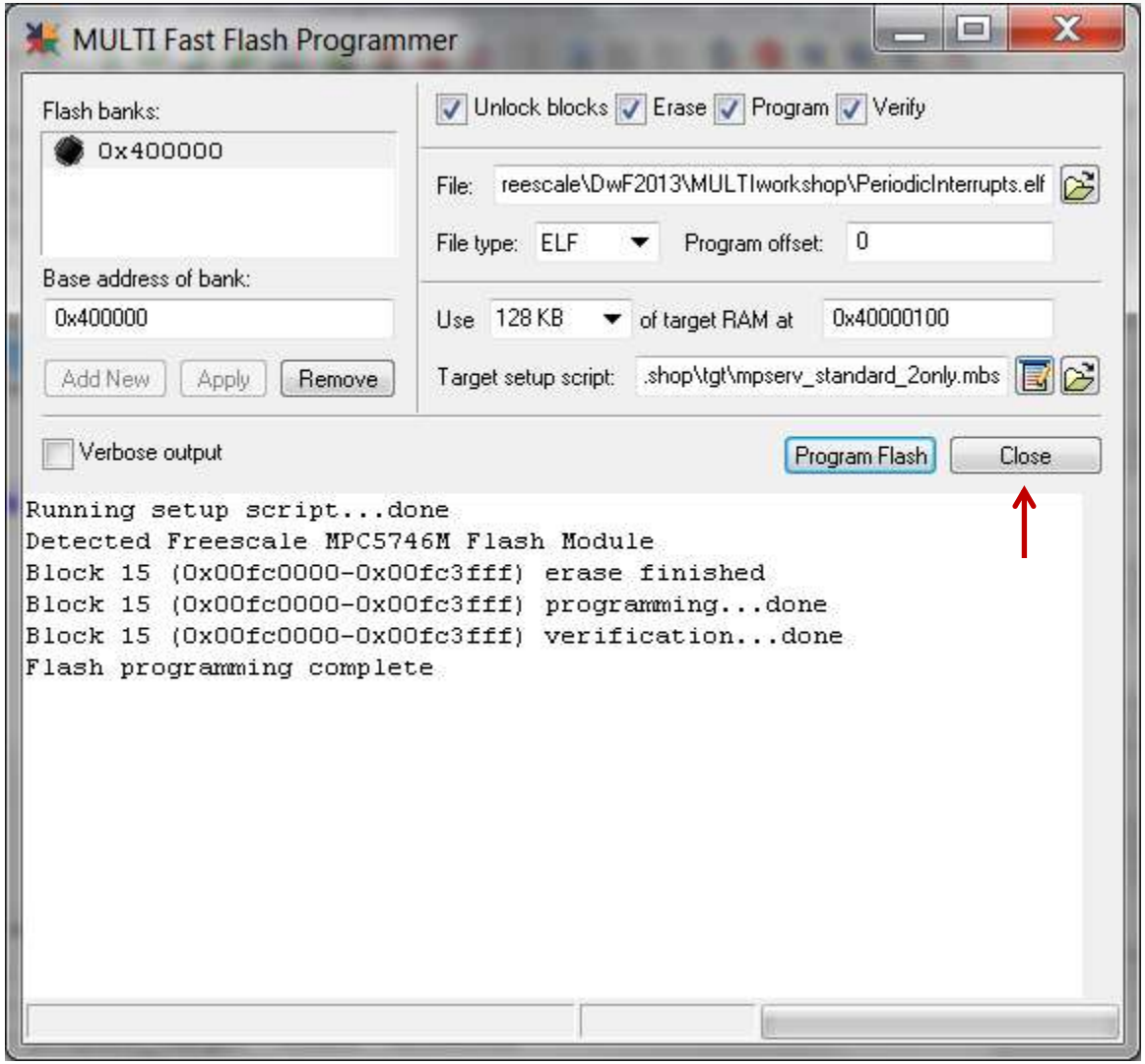
Hands-on 4.4: **Program Target Flash Memory - 2**

- Make sure the **base address** is **0x400000** (will be remembered once set)
- Make sure **Unlock blocks** is **checked** (will be remembered once set)
- Verify that the “2only” setup script is specified
- **Left click** on the “**Program Flash**” button



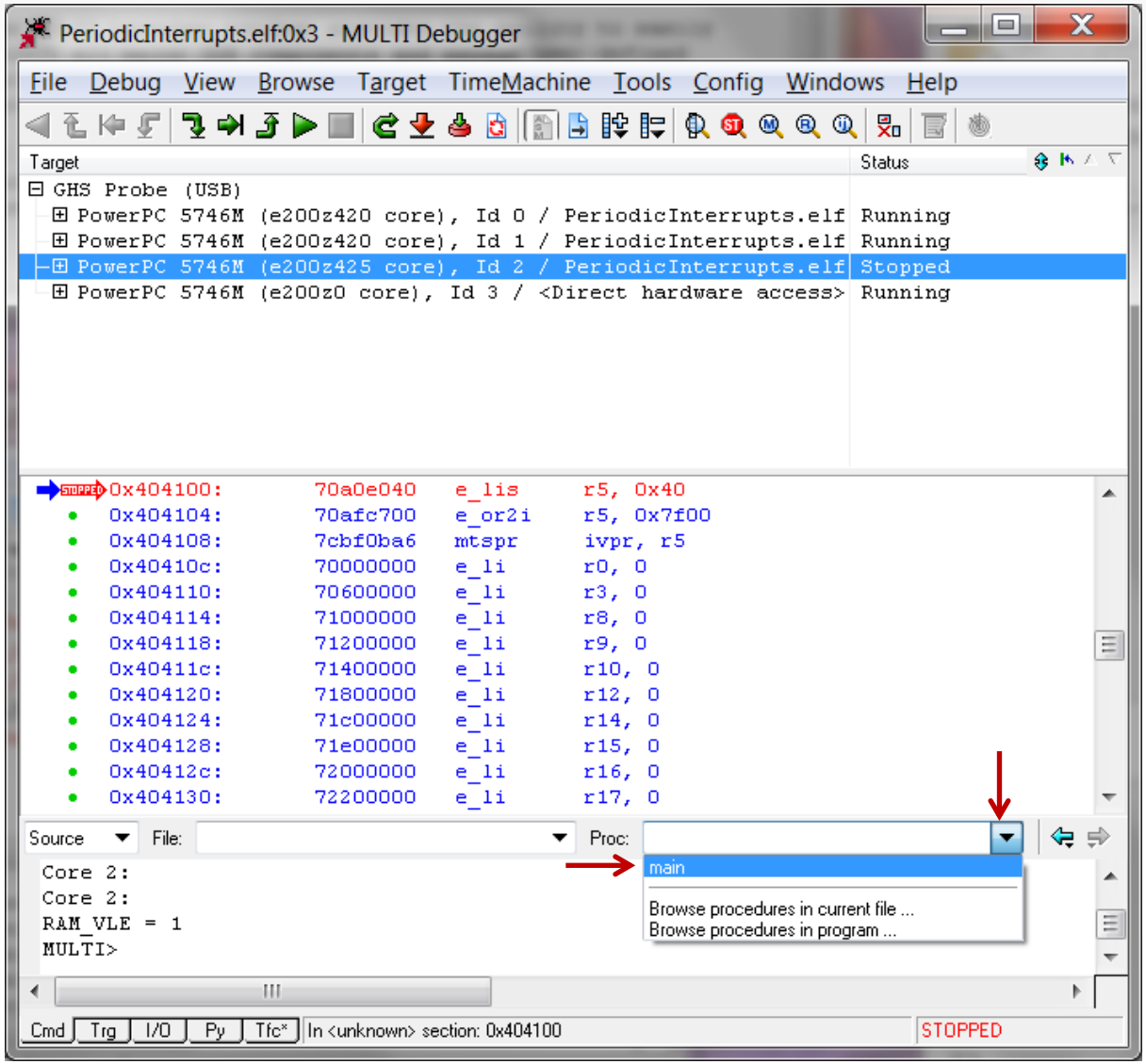
Hands-on 4.4: **Program Target Flash Memory - 3**

- If there is an error, make sure that the target is powered
- **Left click** on the "**Close**" button



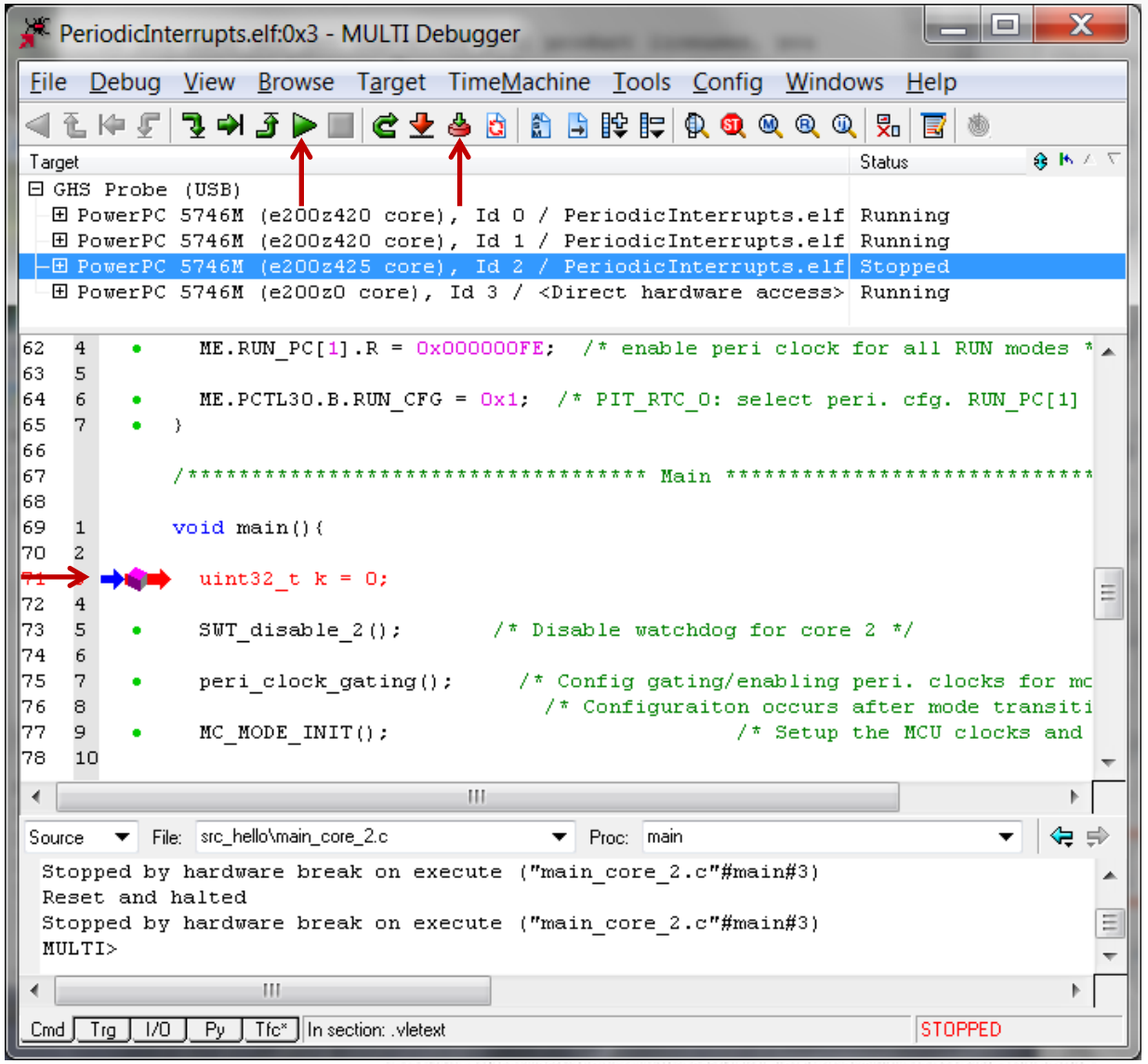
Hands-on 4.5: **Navigate to Main**

- **Left click** on the "**Proc**" pull down menu
- **Left click** on the "**main**" entry



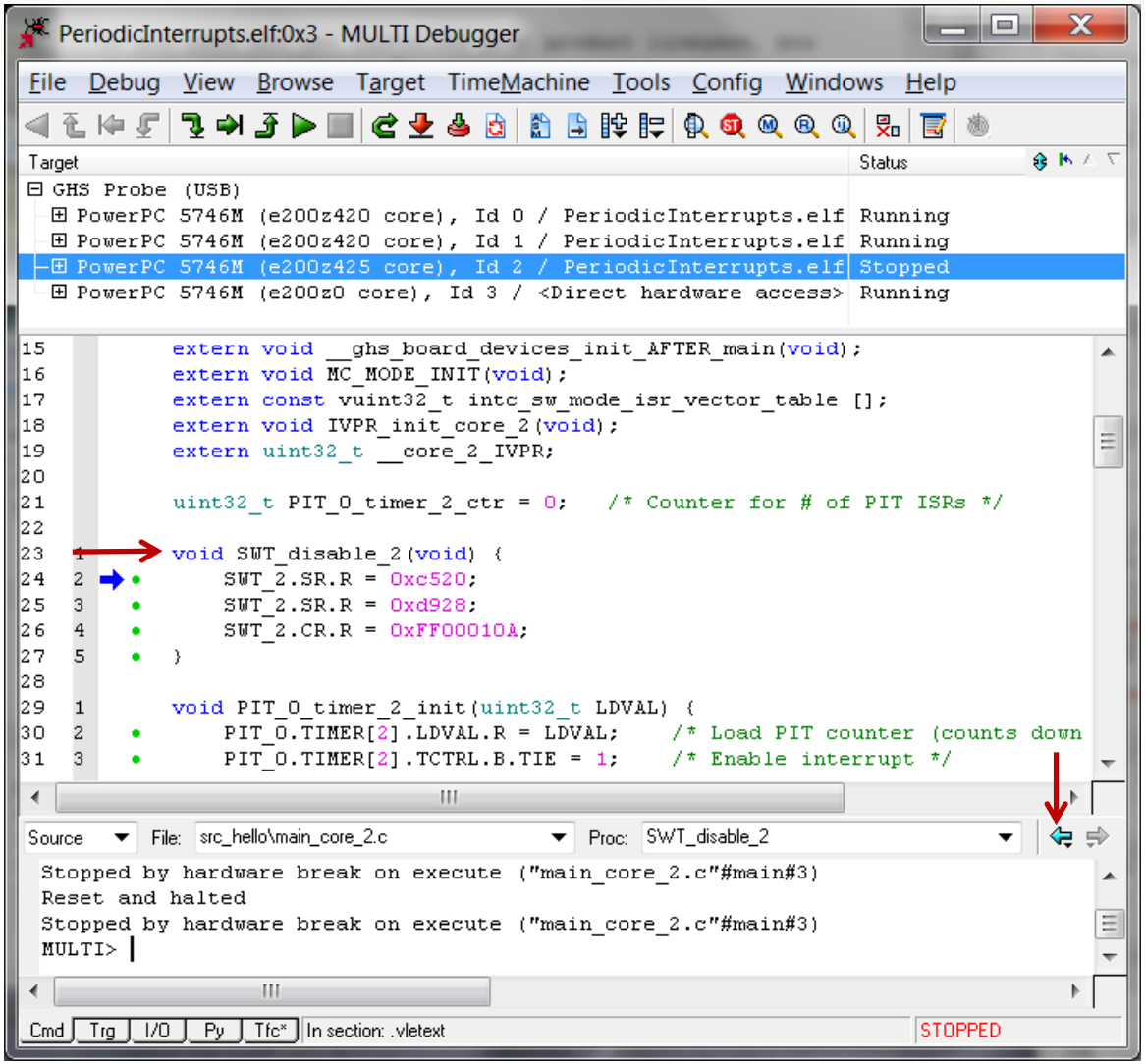
Hands-on 4.6: **Run to a Hardware Breakpoint**

- **Left click** on the **green dot** at main:3 to set a hardware breakpoint
- **Left click** on the "**Go**" button
- Press "**Reset**" button and "**Go**" button to run again
- **Left click** on the breakpoint at main:3 to remove it



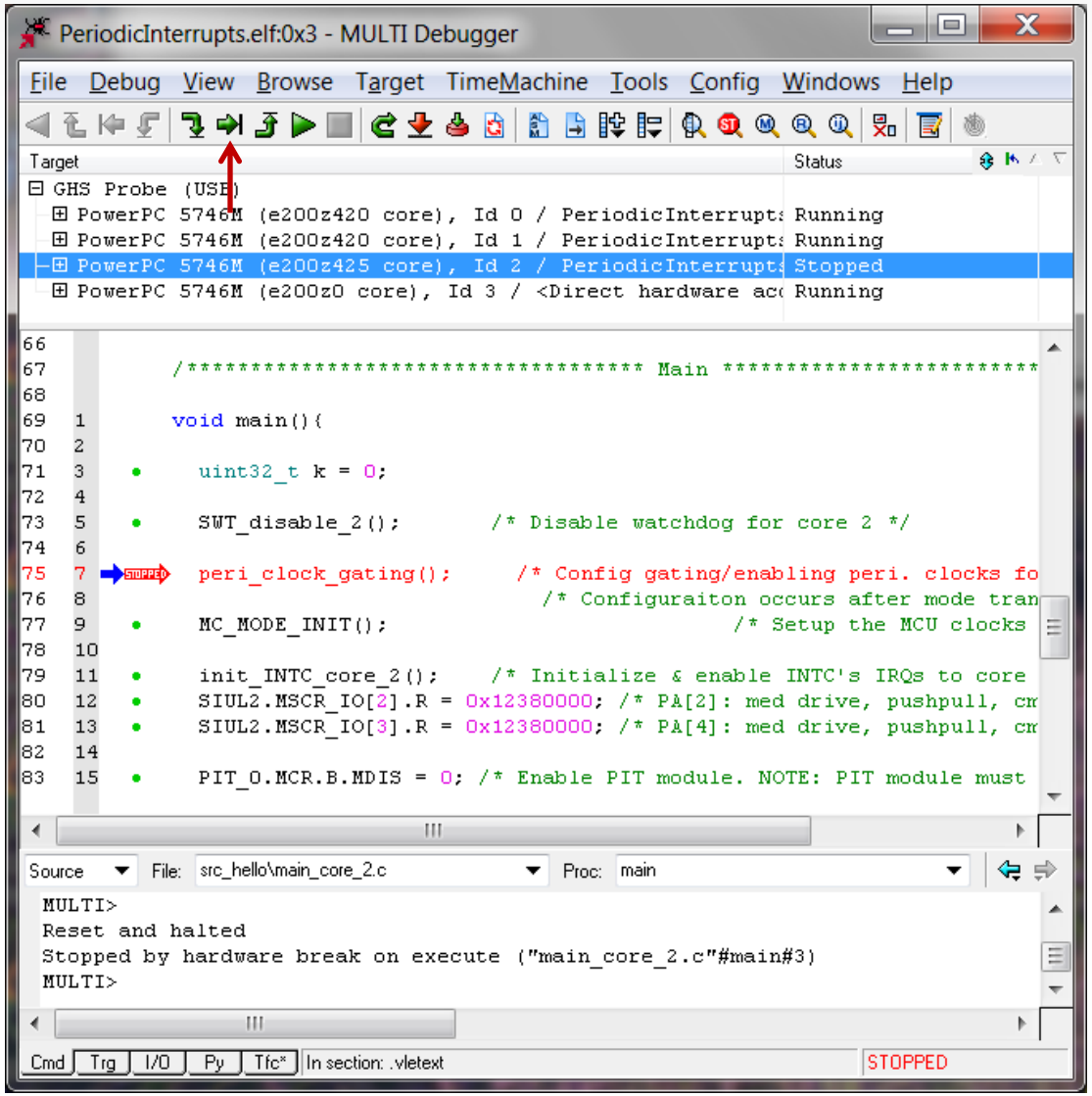
Hands-on 4.7: **Go to Definition**

- **Right click** on **SWT_disable_2()**
- **Left click** on "**Go To Definition**" in pop-up menu
- Briefly **Left click** on the **left arrow** in the lower right to return to the prior view of main



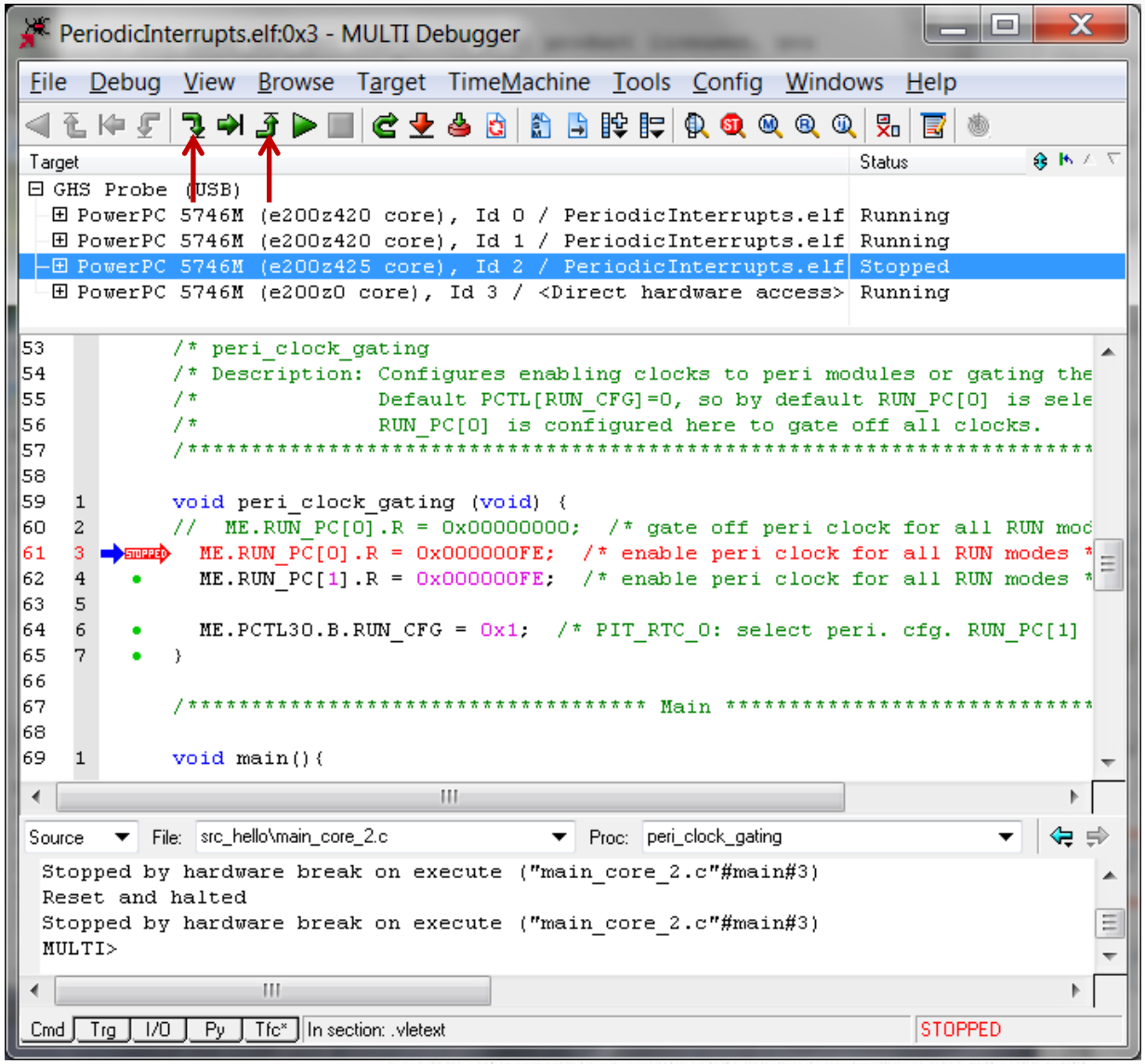
Hands-on 4.8: **Step Over**

- **Left click** the "**Next**" button twice to step over SWT_disable_2



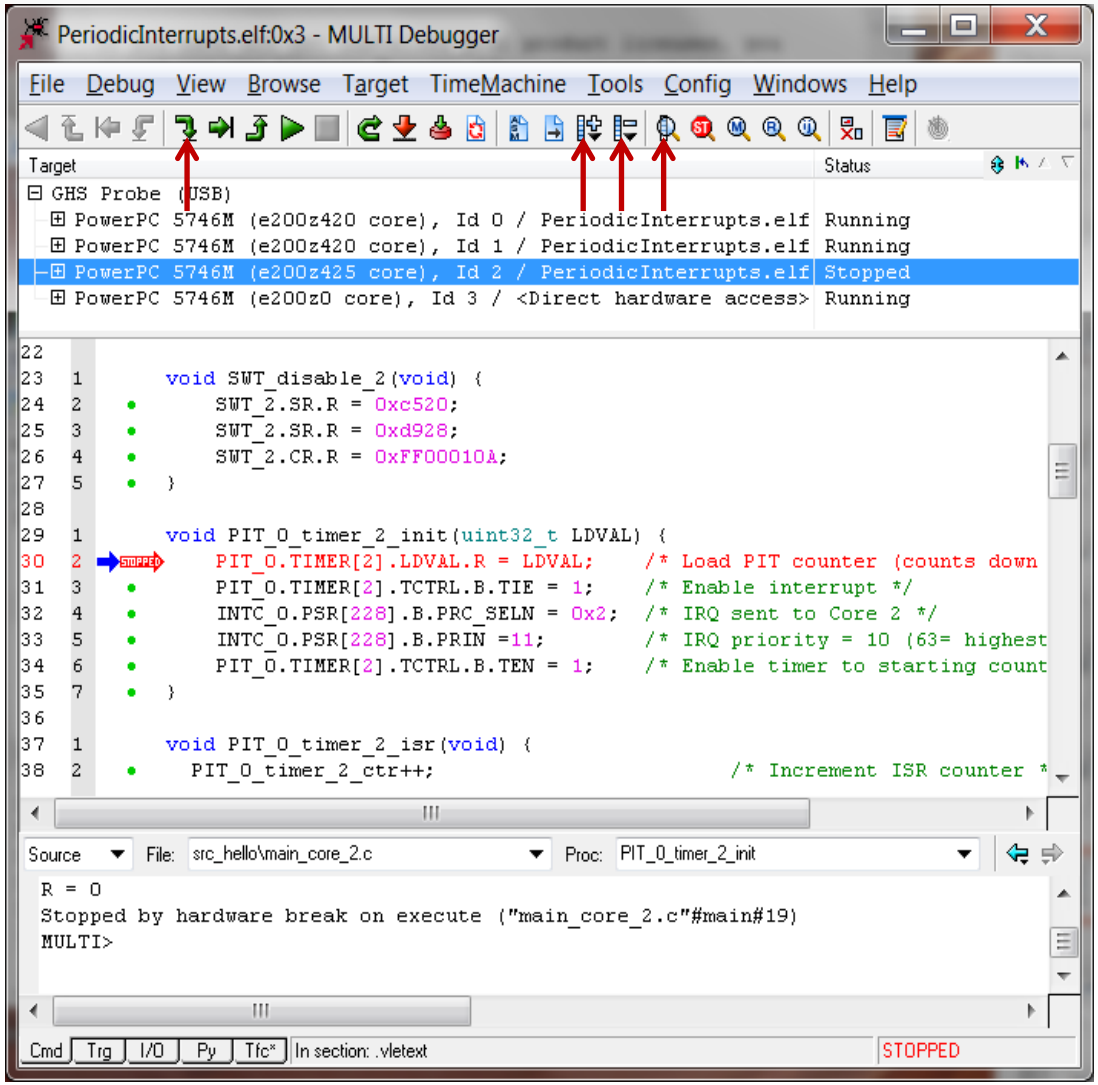
Hands-on 4.9: **Step In and Return**

- **Left click** on the "**Step**" button to step into peri_clock_gating
- **Left click** on the "**Return**" button to run back to Main



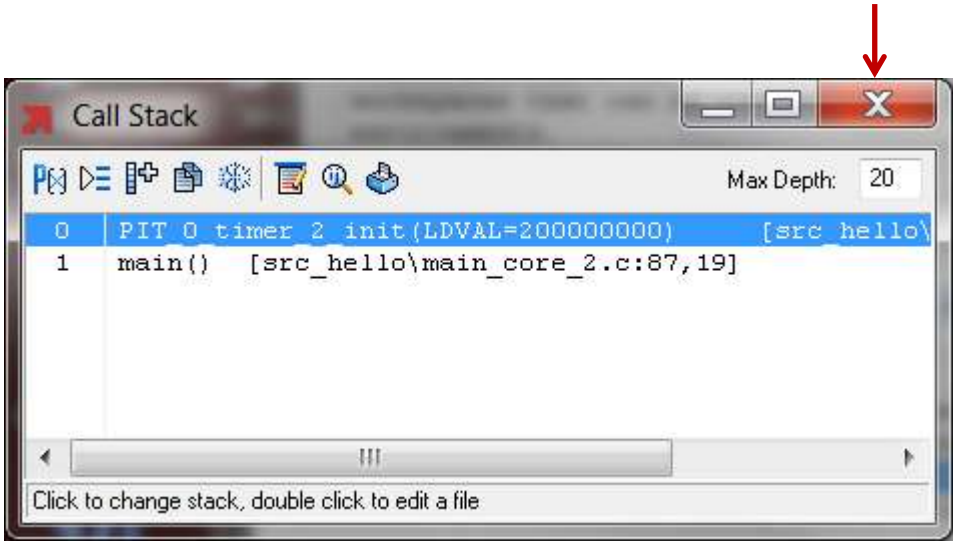
Hands-on 4.10: **Navigate the Call Stack**

- Set a breakpoint on Main:19, run to it, and then remove the breakpoint
- **Left click** on "**Step**" to step into PIT_0_timer_2_init
- Left click on the "**Upstack**" button to show the caller (Main)
- **Left click** on the "**Downstack**" button to return to PIT_0_timer_2_init
- **Left click** on the "**Call Stack**" button to open the call stack window



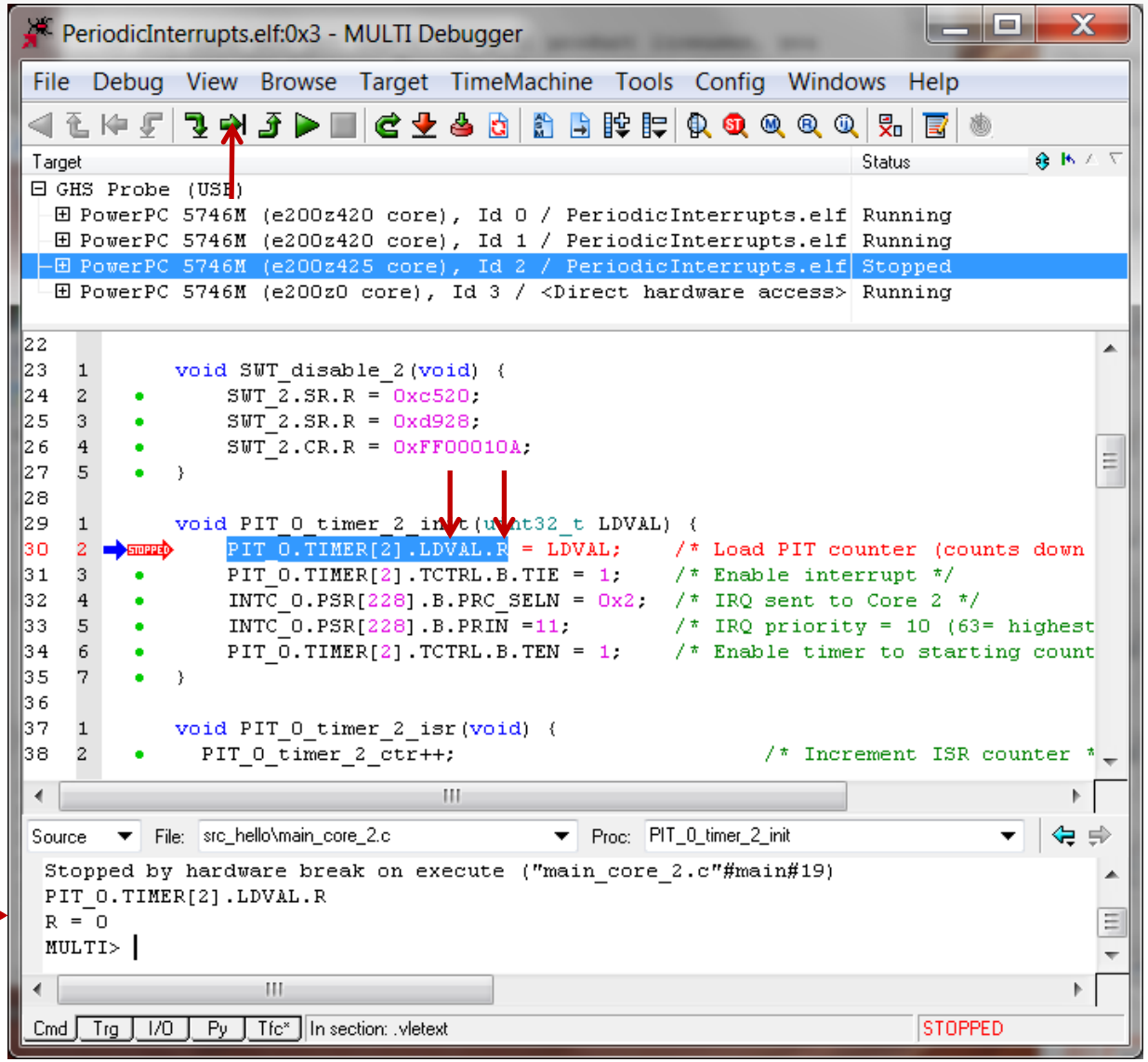
Hands-on 4.11: **View the Call Stack Window**

- Note the stack entries
- **Close** the call stack window



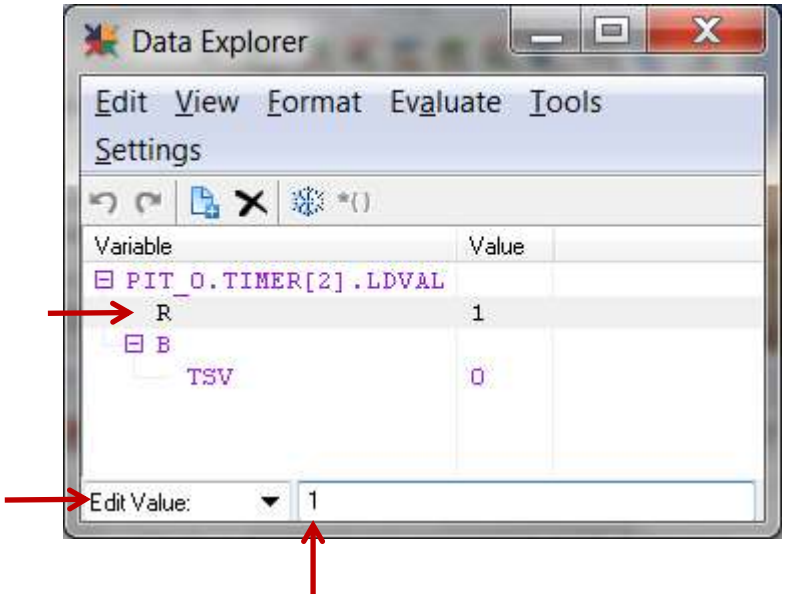
Hands-on 4.12: **View Structured Element in Cmd Pane**

- **Left click** on **R** in **PIT_0.TIMER[2].LDVAL.R** to display its value in the Cmd pane
- **Double click** on "**LDVAL**" in **PIT_0.TIMER[2].LDVAL.R** to display the element in the data explorer window



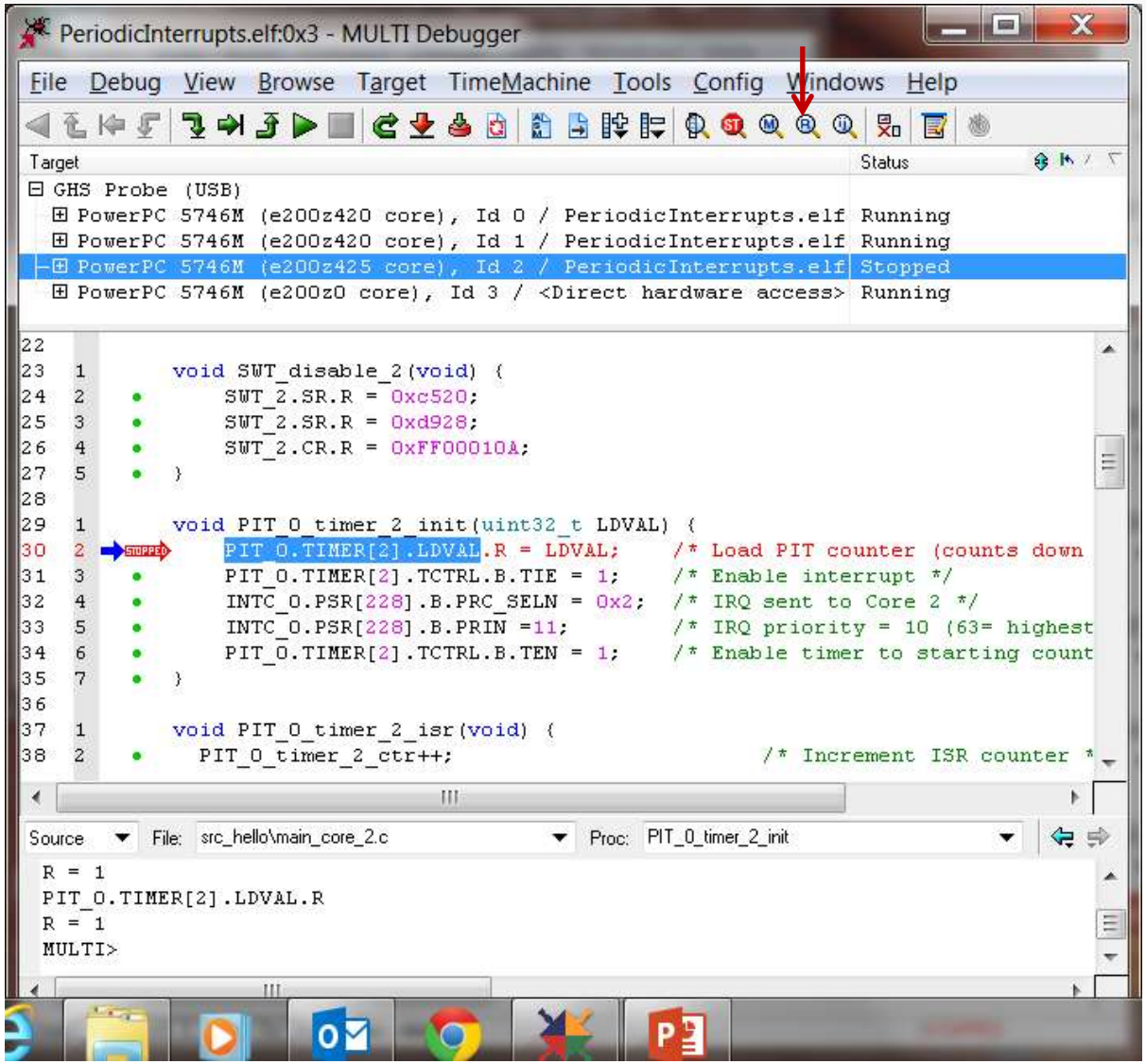
Hands-on 4.13: **Change a Structure Element**

- Expand the element “B” in the Data Explorer window
- Left click on the “**R**” in the Data Explorer window
- **Select** “**Edit Value**” in the action pull down menu
- Type a **1** and press the **enter** key



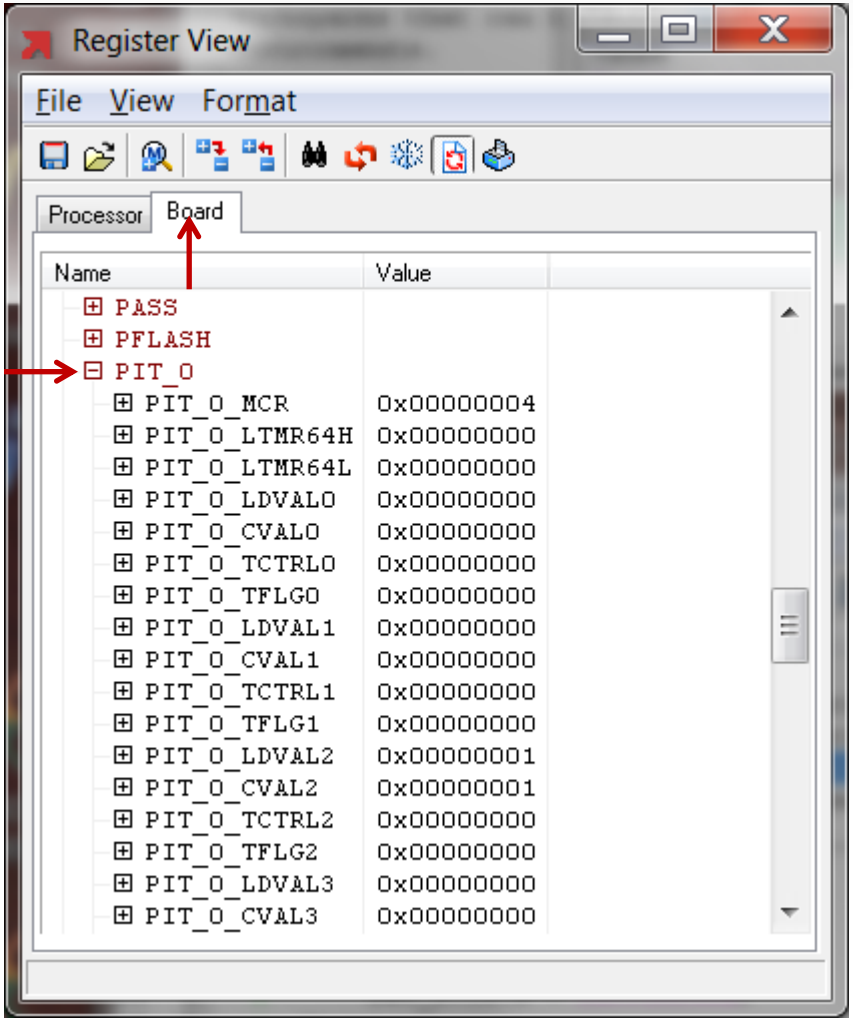
Hands-on 4.14: **Display a Peripheral Register - 1**

- **Left click** on the "**Registers**" button in the Debug window



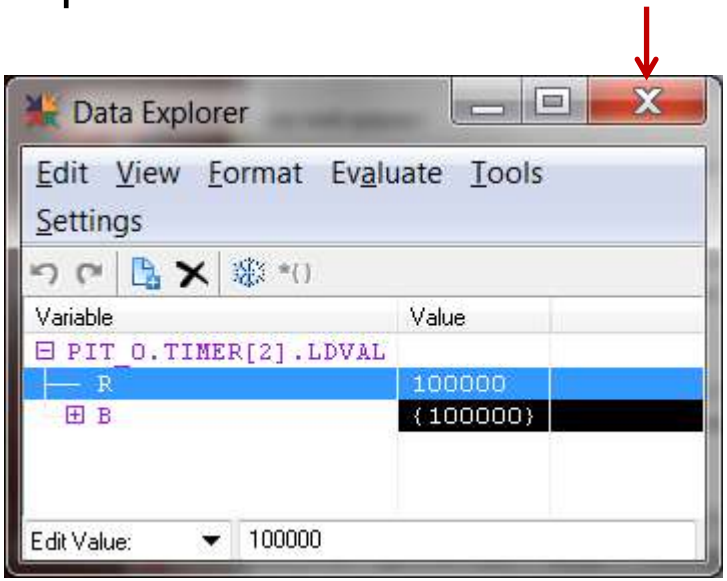
Hands-on 4.14: **Display a Peripheral Register - 2**

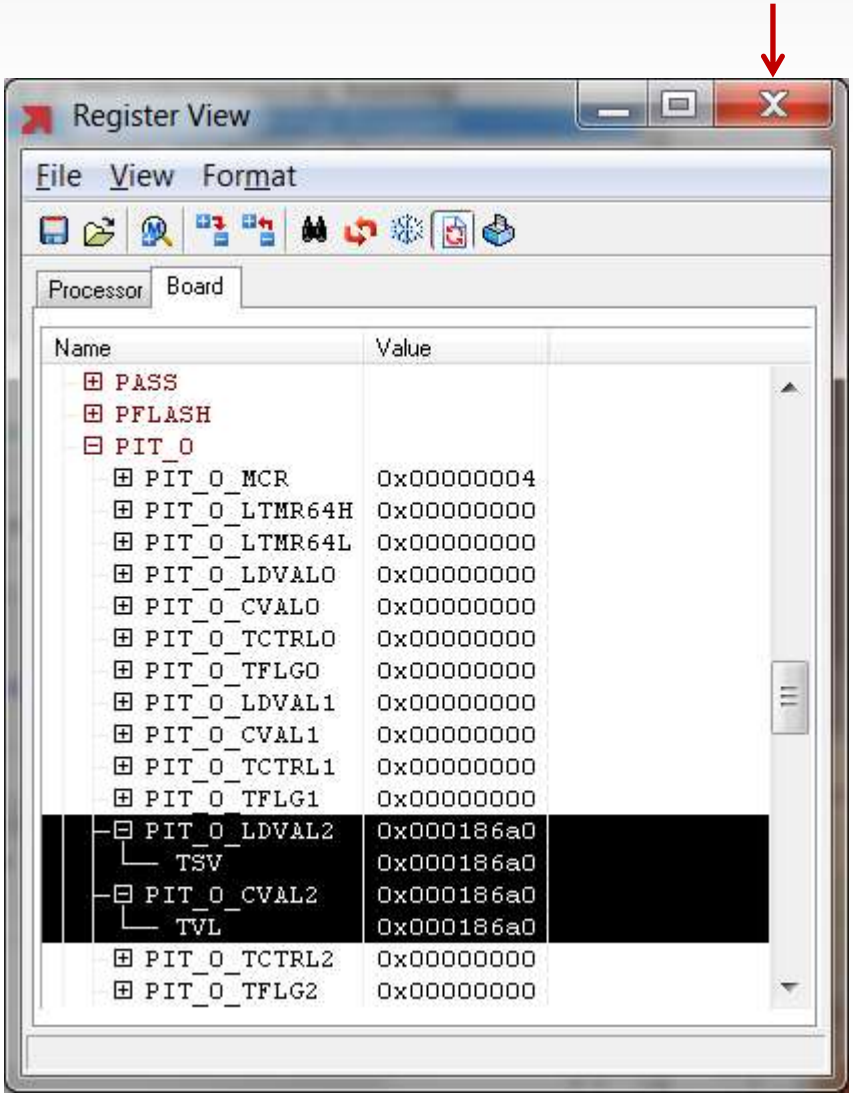
- **Left click** on the "**Board**" pane in the register window
- **Scroll down to PIT_0** and expand it



Hands-on 4.14: **Display a Peripheral Register - 3**

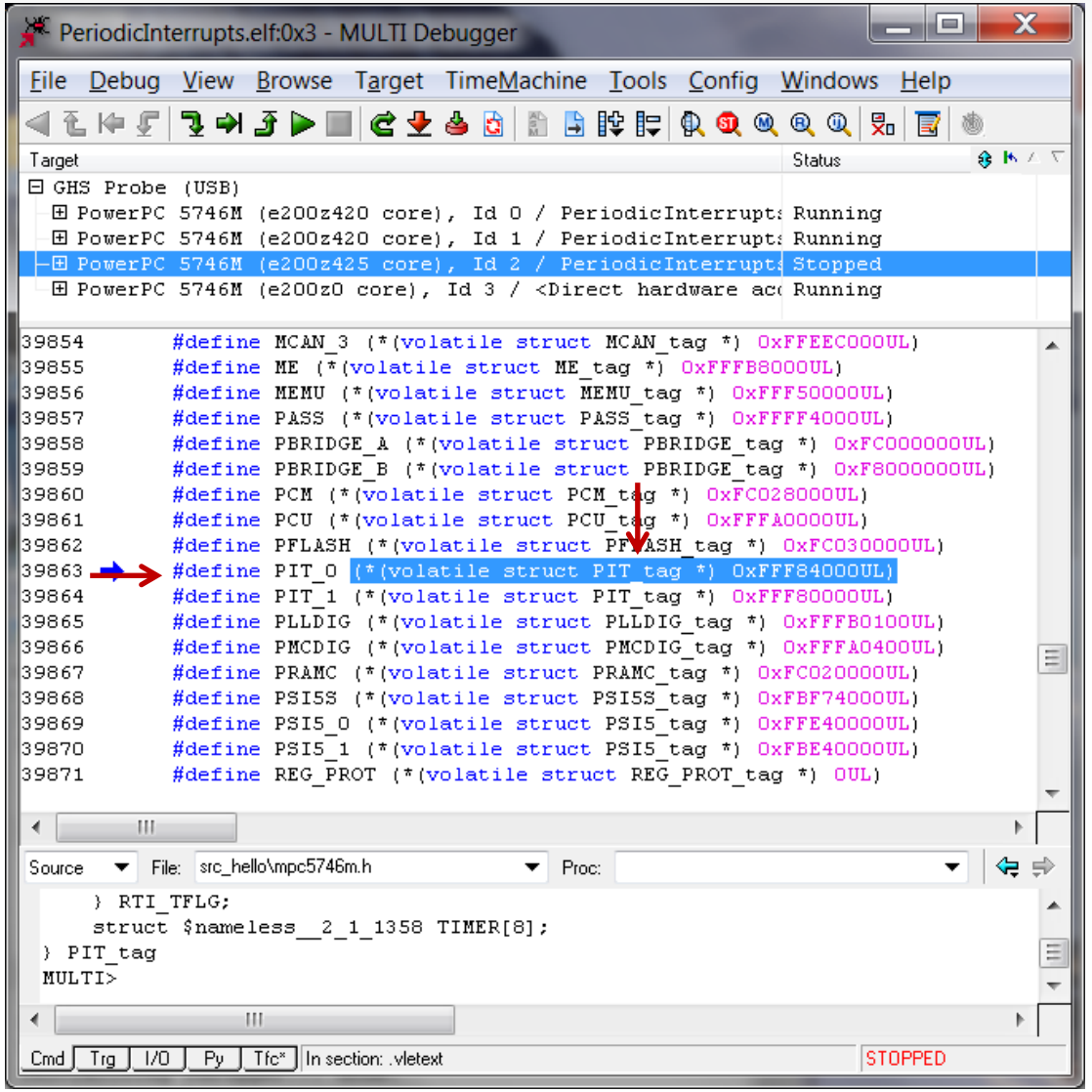
- **Left click** on the "**step**" button in the debug window
- Note the update in the Register window and Data Explorer window
- **Close** the Register and Data Explorer windows





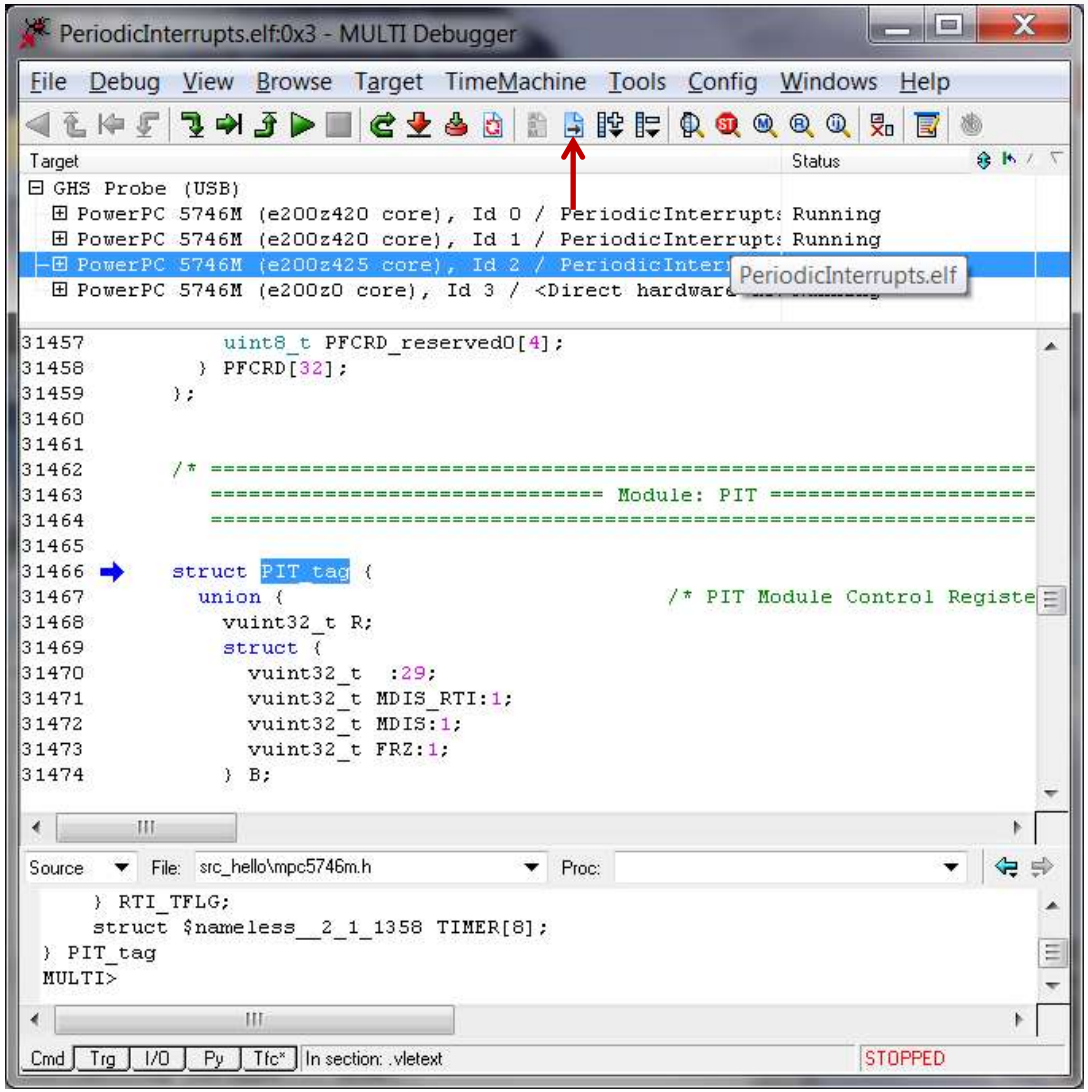
Hands-on 4.15: **Display Structure Definition - 1**

- **Right click** on the **PIT_0** in **PIT_0.TIMER[2].LDVAL.R** in the Debugger window
- **Left click** on "**Go To Definition**" in the pop-up window
- The location where the PIT macro is defined is displayed
- **Left click** on "**PIT_tag**" to display where the structure is defined



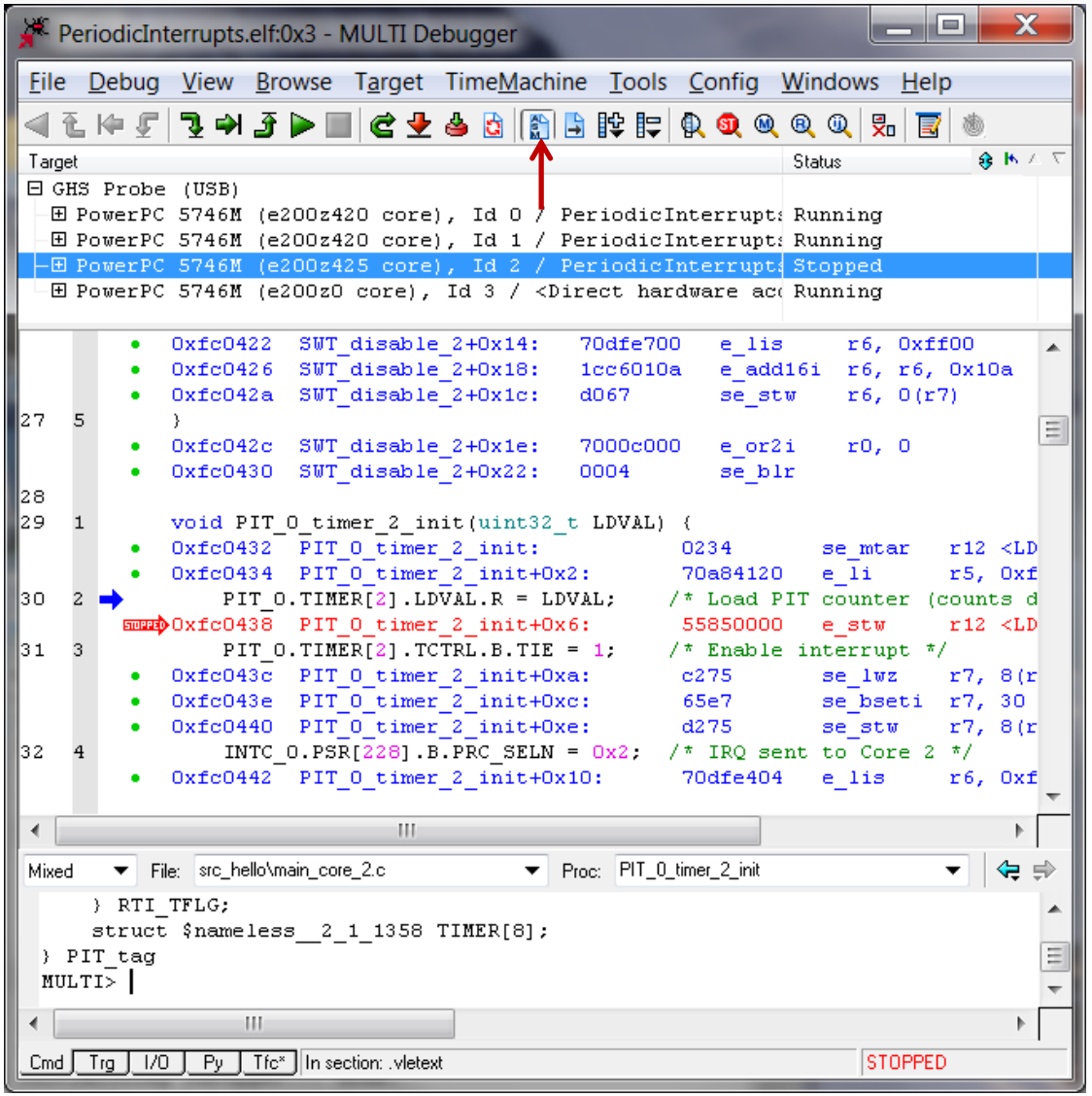
Hands-on 4.15: **Display Structure Definition - 2**

- **Left click** on the "**program counter**" button to return to current execution point



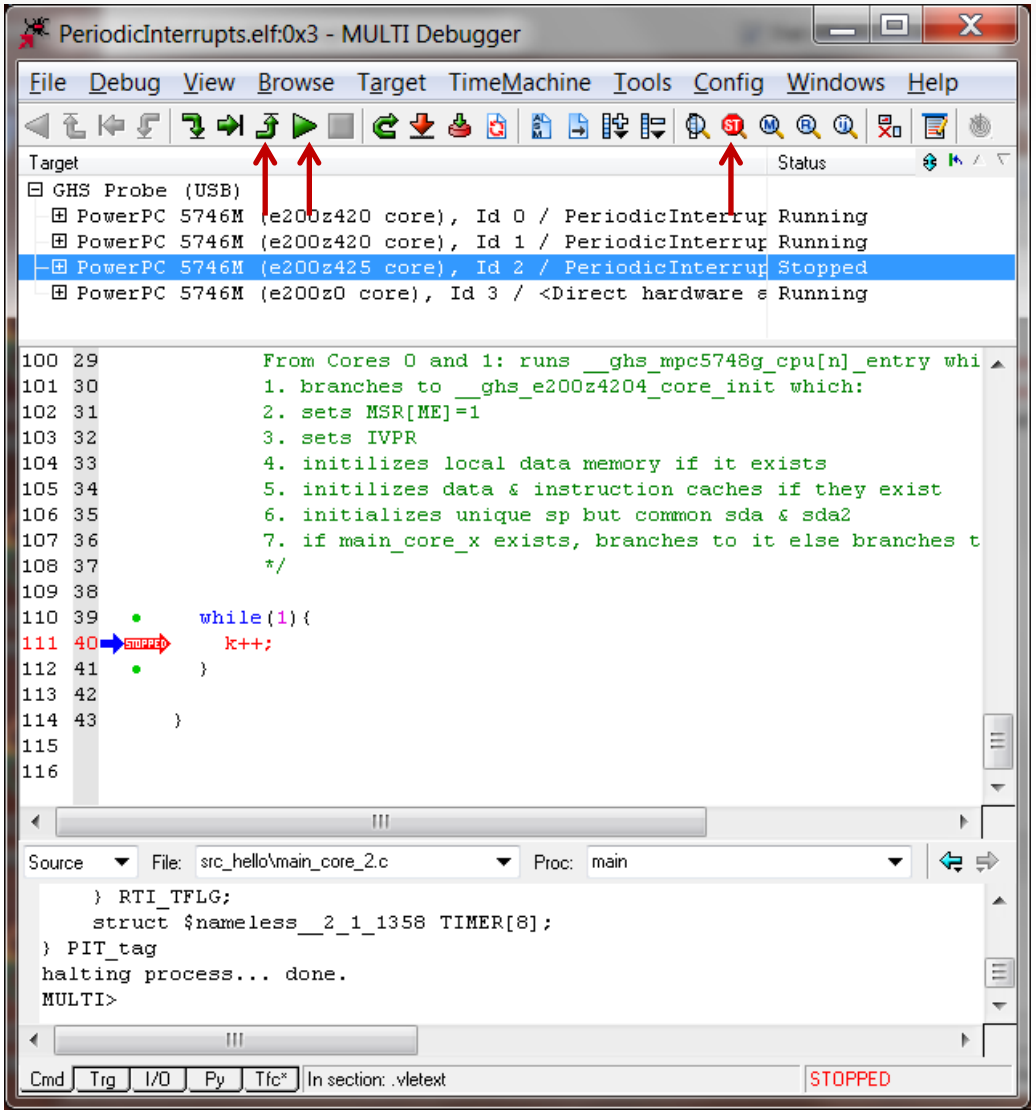
Hands-on 4.16: **Show in Mixed Mode**

- **Left click** on the "**Assembly**" button to get mixed display
- **Left click** on the "**Assembly**" button again to return to source display



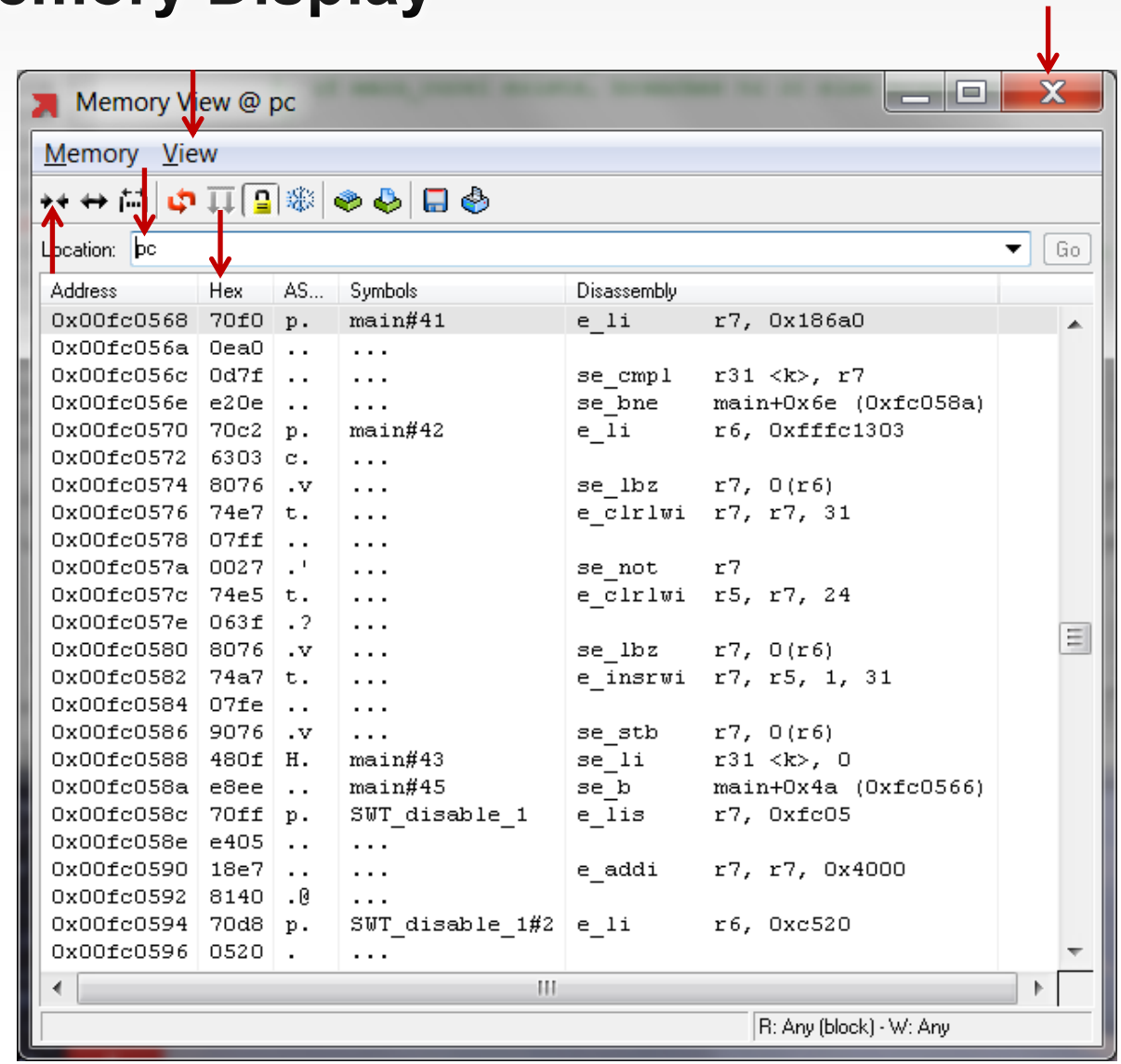
Hands-on 4.17: **Run and Halt Execution**

- **Left click** on the "**Go**" button
- Note flashing LEDs (jumpers must be installed)
  - interrupts are occurring
- **Left click** on the "**Halt**" button
- **Left click** on the "**Memory**" button



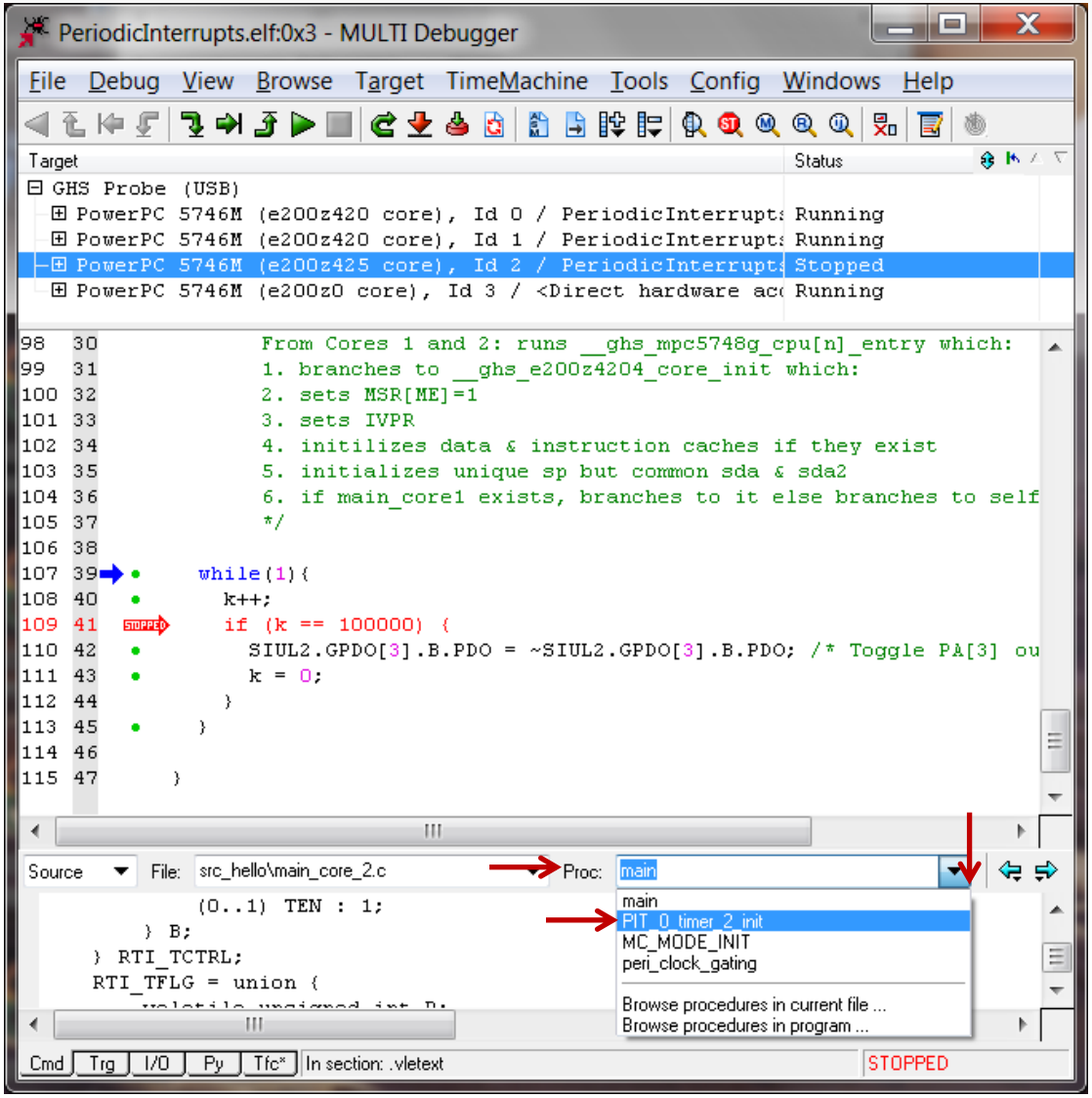
Hands-on 4.18: **Memory Display**

- Type “**pc**” in the location box of the memory window and press enter
- **Right click** on “Hex” and Left click on 2 bytes in the pop-up menu
- **Left click** on the “**contract**” button twice
- **Left click** on **View->Add New Column-> Disassembly** menu item
- **Close** the memory window



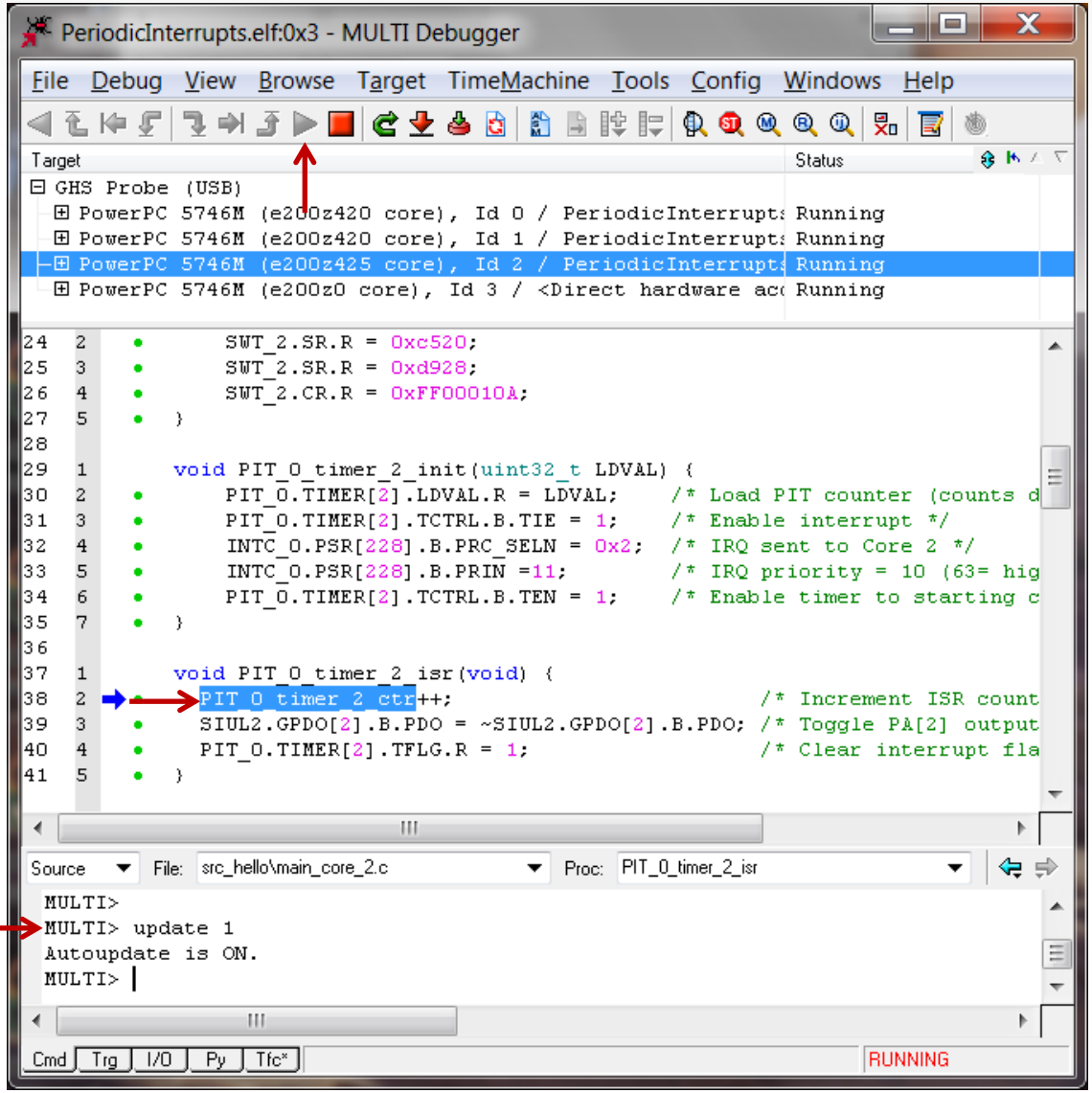
Hands-on 4.19: **Display Variables While Running - 1**

- **Left click** on the Debugger's "**Proc**" pull down menu
- **Left click** on the "**PIT_0_timer_2_init**" entry



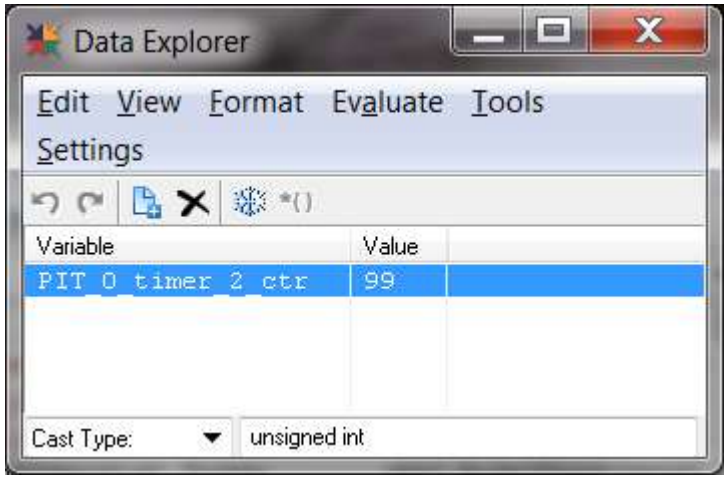
Hands-on 4.19: **Display Variables While Running - 2**

- **Scroll** down and **double click** on **PIT_0_timer_2_ctr**
- **Left click** on "**Go**" button
- Enter "**update 1**" in the debugger Cmd pane



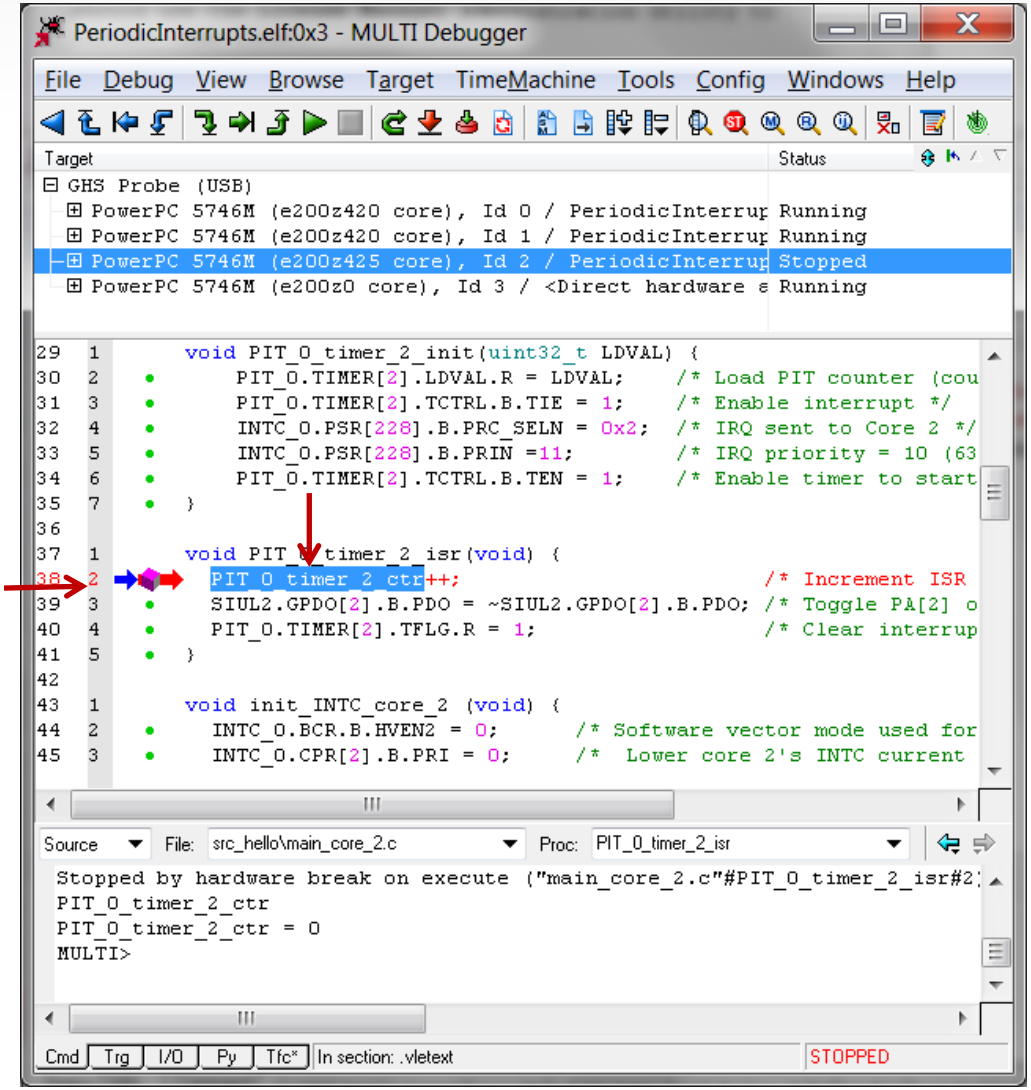
Hands-on 4.19: **Display Variables While Running - 3**

- Note that the variable is changing in the Data Explorer window
- Enter "**update 0**" in the debugger Cmd pane
- The variable stops changing
- **Close** the Data Explorer window



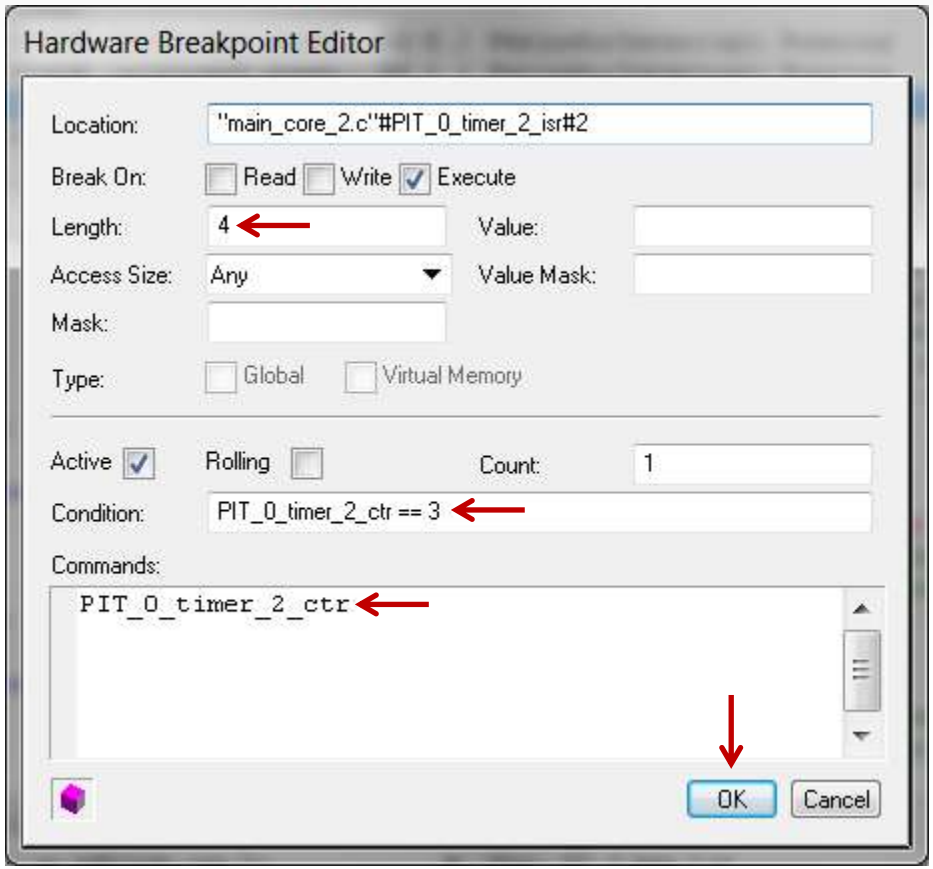
Hands-on 4.20: **Use a Complex Breakpoint - 1**

- Set a breakpoint at PIT_0_timer_2_isr:2 by left clicking on the **green dot**
- Note that the program stops there
- **Left click** on PIT_0_timer_2_ctr and copy it (cntl-c)
- **Right click** on the **breakpoint**
- **Left click** on "**Edit Hardware breakpoint…**" in the popup menu



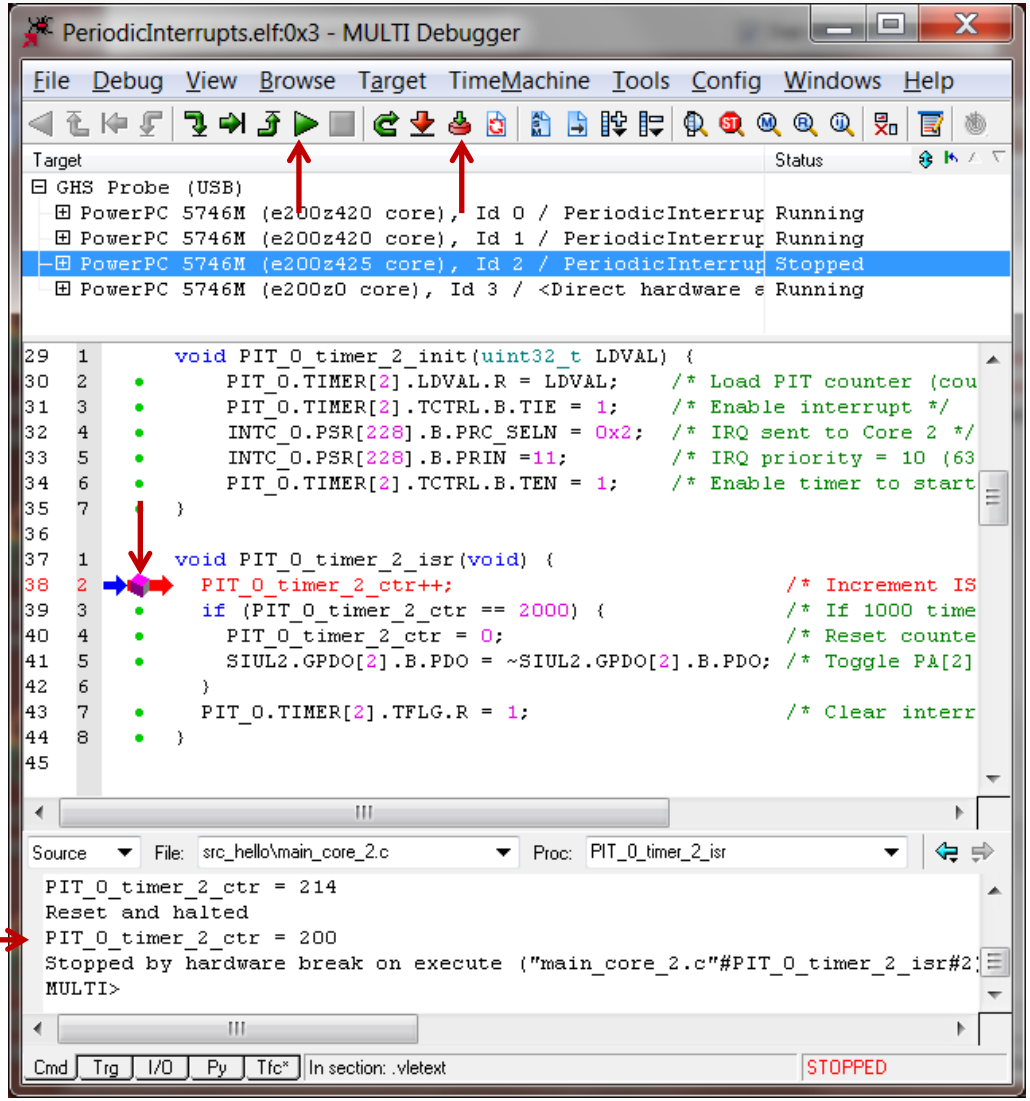
Hands-on 4.20: **Use a Complex Breakpoint - 2**

- Paste (cntl-v) **“PIT_0_timer_2_ctr”** into the Condition box of the hardware breakpoint window and then type in “**== 200”**
- Verify that a 4 is in the Length box
- Paste “**PIT_0_timer_2_ctr**” in the Commands box
- **Left click** on the **OK** button



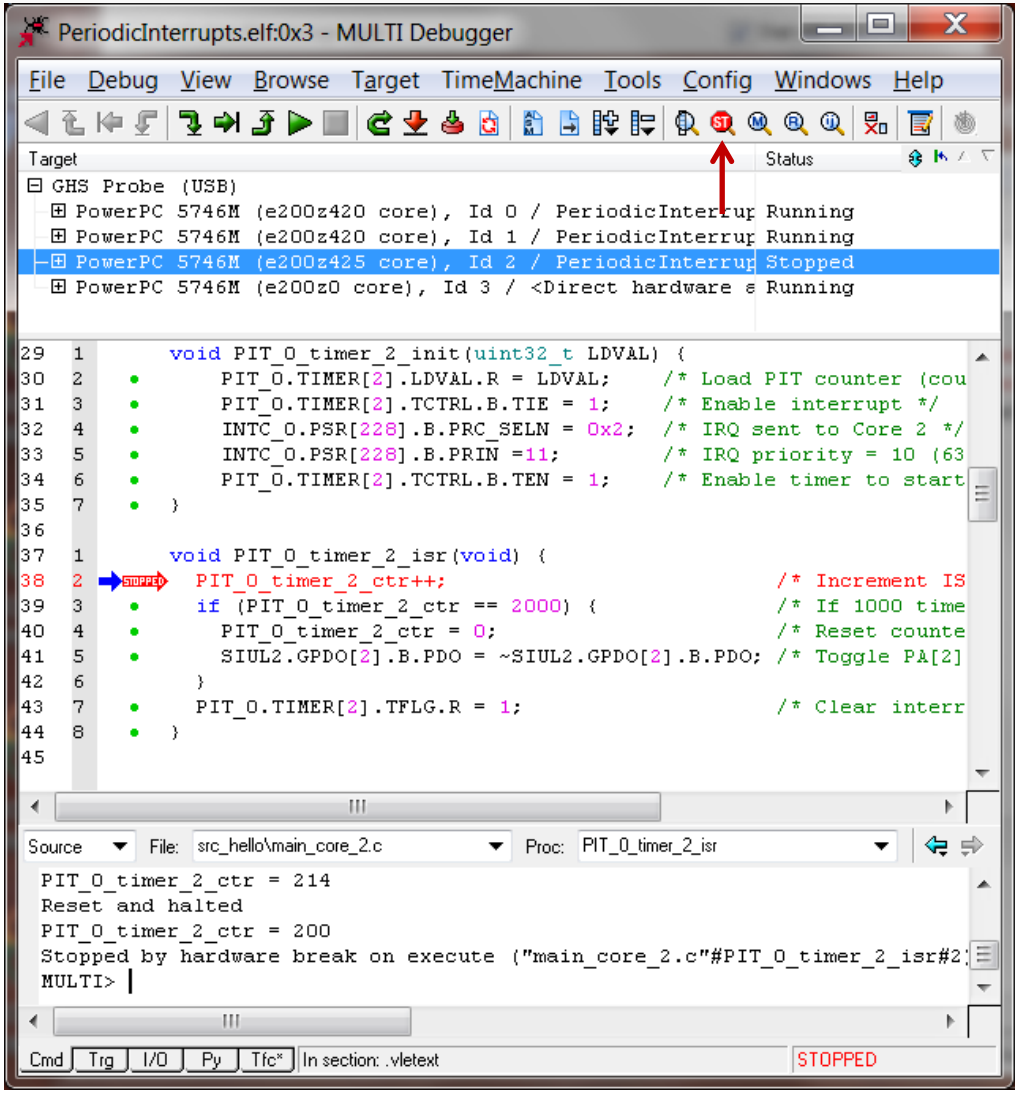
Hands-on 4.20: **Use a Complex Breakpoint - 3**

- **Left click** on the "**Reset**" button and then the "**Go**" button in the Debugger window
- Wait for a few seconds
- The processor stops when PIT_0_timer_2_ctr reaches 200
- The value of PIT_0_timer_2_ctr is displayed in the Cmd pane
- **Left click** the breakpoint at Pit1ISR:2 to remove it



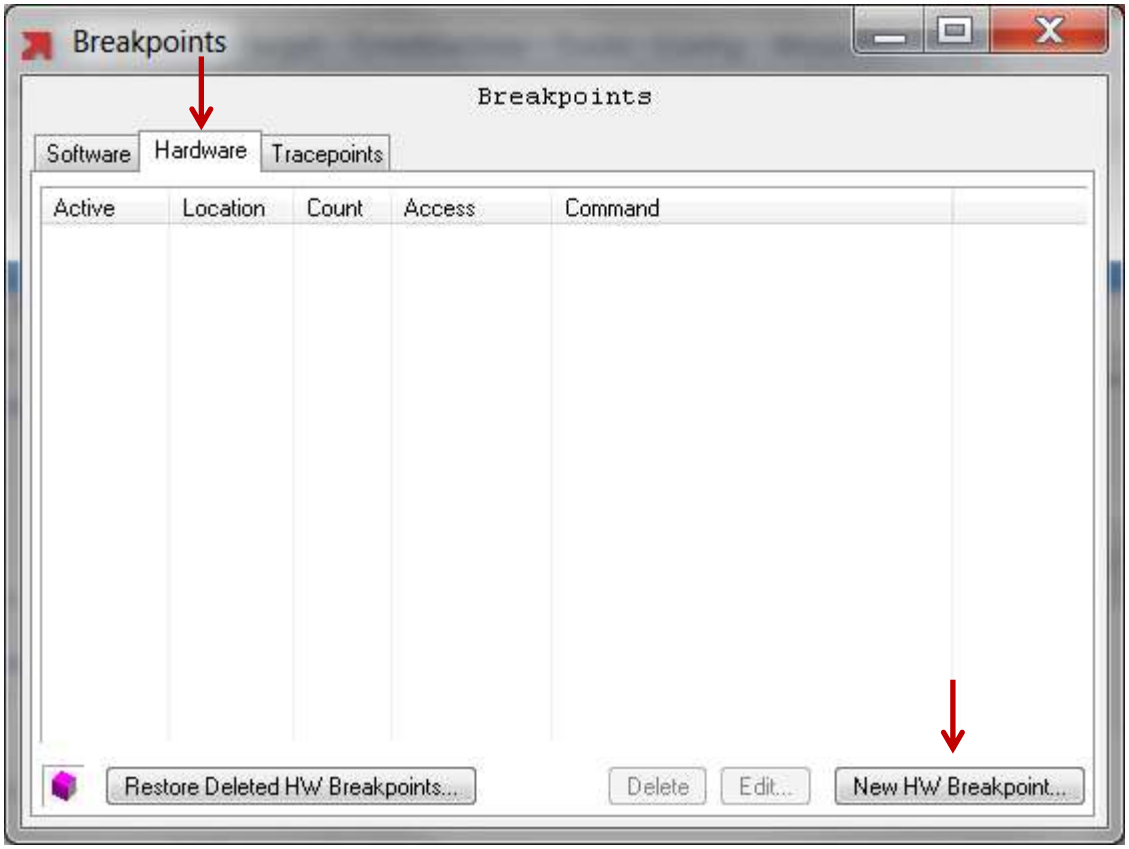
Hands-on 4.21: **Use a Data Breakpoint - 1**

- **Left click** on the "**Breakpoints**" button in the Debugger window



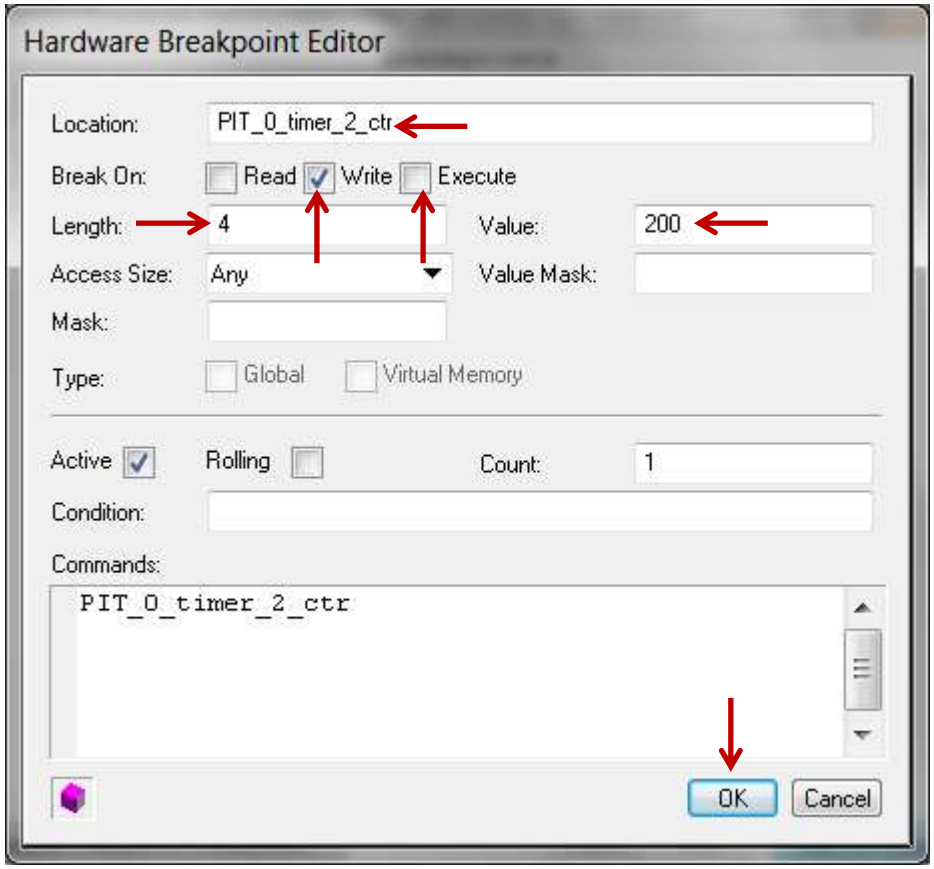
Hands-on 4.21: **Use a Data Breakpoint - 2**

- **Left click** on the **Hardware tab** of the Breakpoints window
- **Left click** on the **"New HW breakpoint" button**



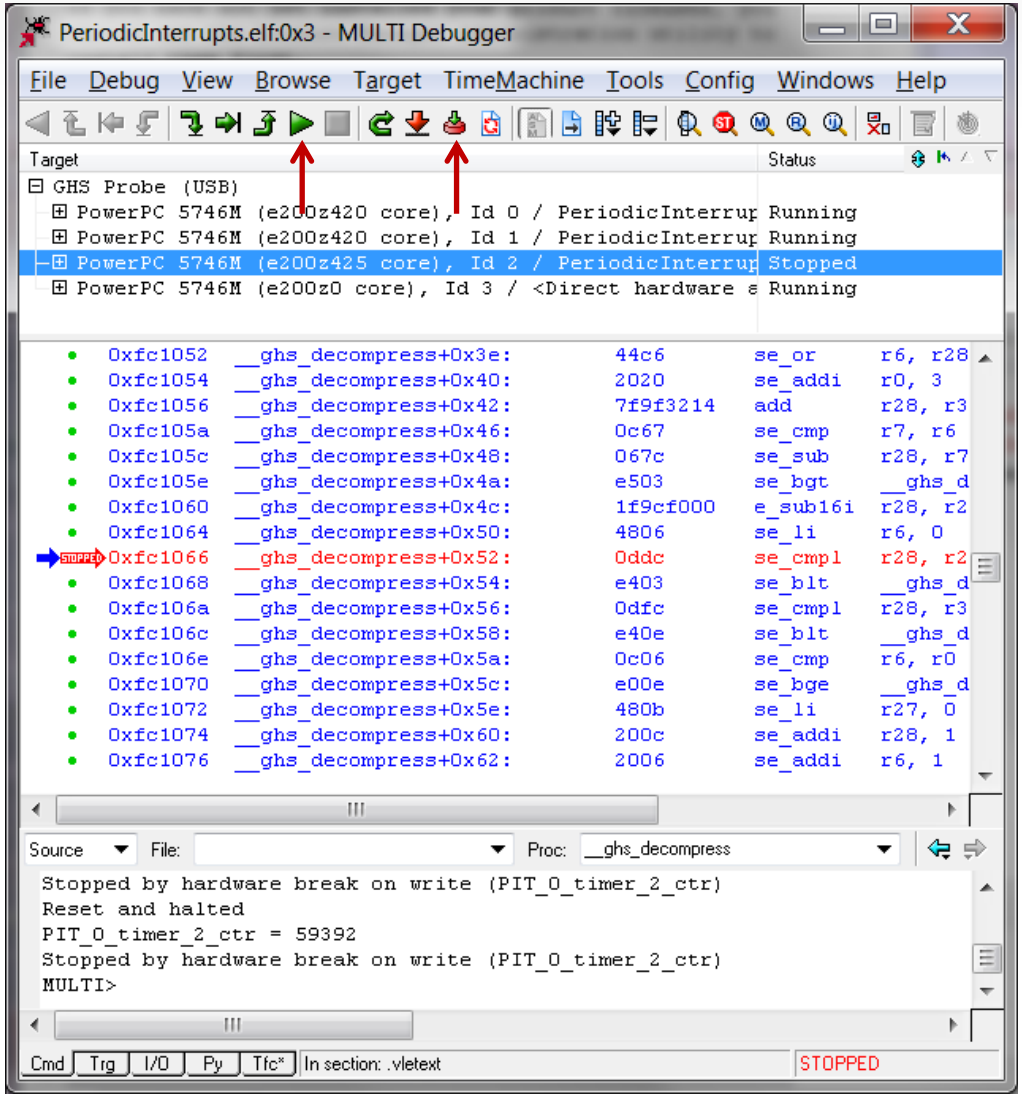
Hands-on 4.21: **Use a Data Breakpoint - 3**

- Paste **PIT_0_timer_2_ctr** in the Location box
- **Left click** to select **Break on Write**
- **Left click** to deselect **Break on Execute**
- Verify that a 4 is in the Length box
- Enter **200** in the Value box
- Paste **PIT_0_timer_2_ctr** in the Commands box
- **Left click** the "**OK**" button



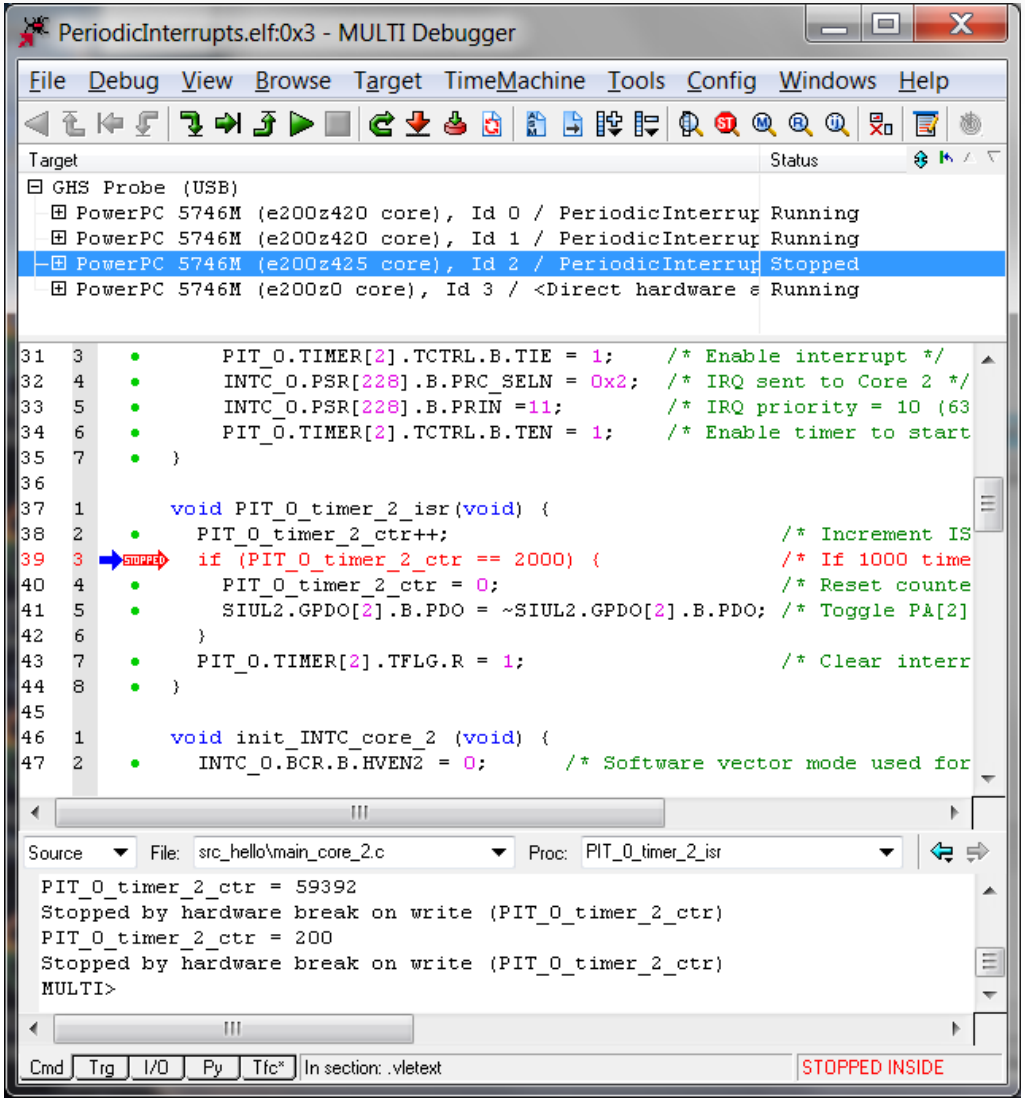
Hands-on 4.21: **Use a Data Breakpoint - 4**

- **Left click** on the "**Reset**" button and then the "**Go**" button in the Debugger window
- A spurious break occurs when the initialized data is copied and decompressed by the startup code.
- **Left click** on the "**Go**" button



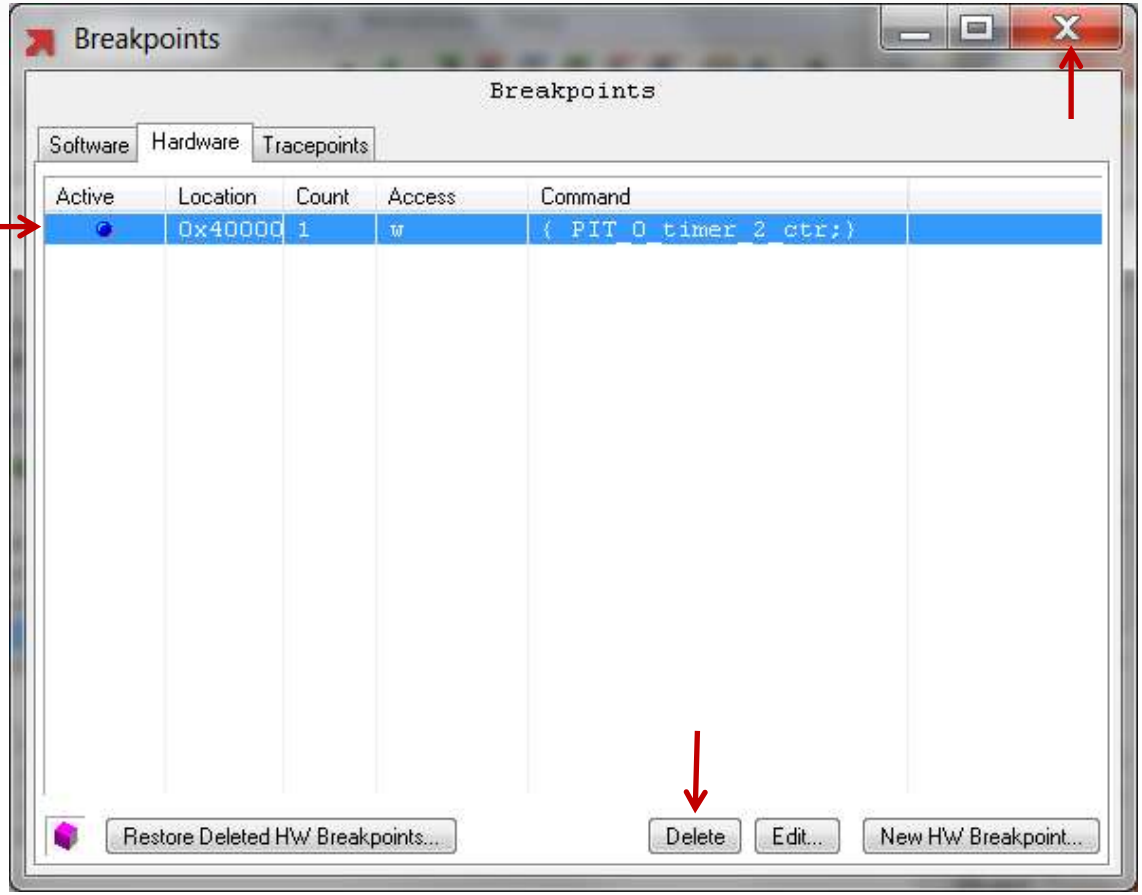
Hands-on 4.21: **Use a Data Breakpoint - 5**

- The processor stops when PIT_0_timer_2_ctr reaches 200
- The value of PIT_0_timer_2_ctr is displayed in the Cmd pane
- Note that the count reached 200 much faster when using the data breakpoint
- Note the effect of the CPU pipeline



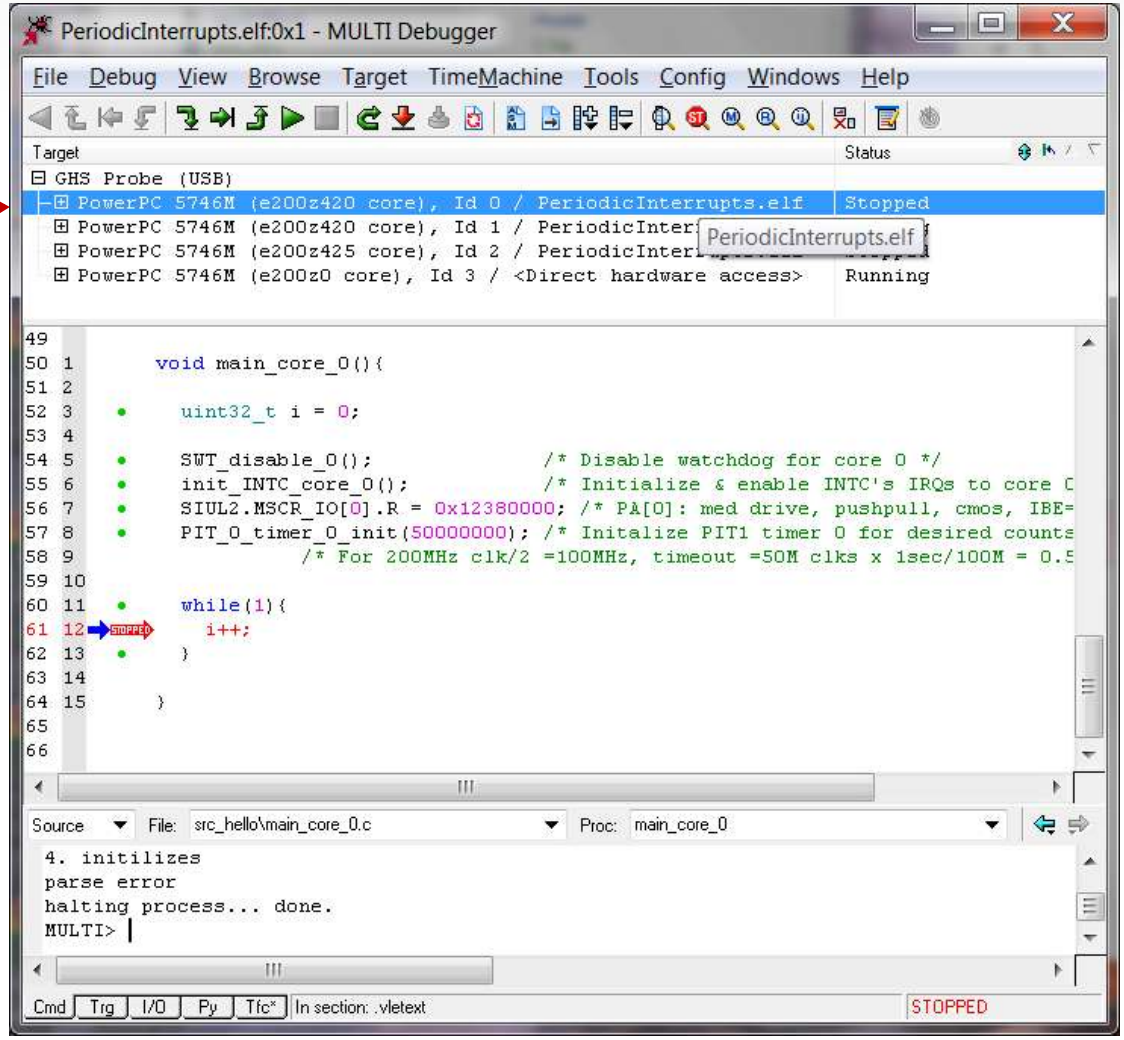
Hands-on 4.21: **Use a Data Breakpoint - 6**

- **Left click** on the hardware breakpoint entry in the Breakpoint window
- **Left click** on the “Delete” button
- Close the Breakpoint window



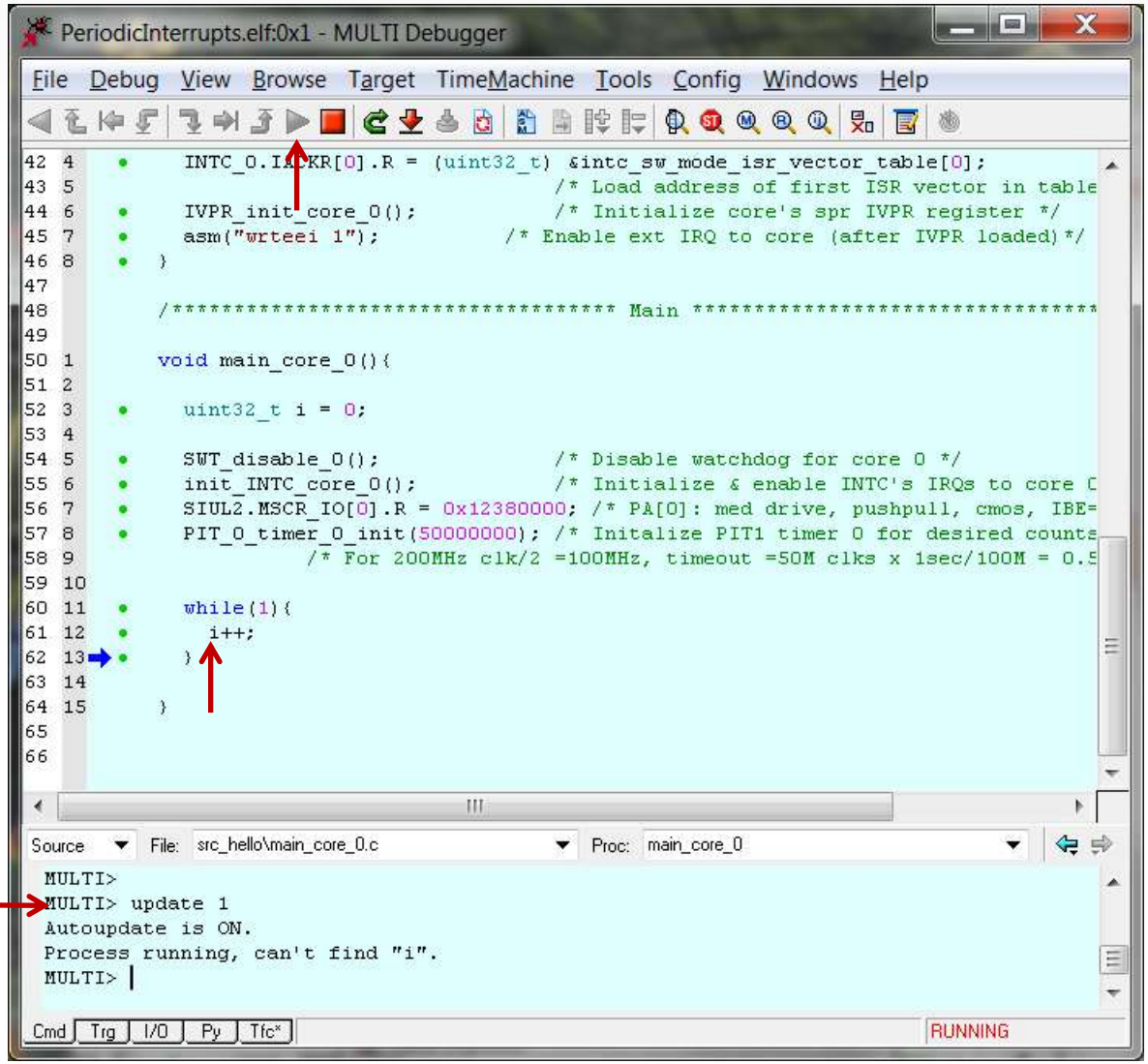
Hands-on 4.22: **Multicore Debugging 1**

- **Left click** on the Probe entry for core ID 0 and you will see that it is running
- **Left Click** on the "**Halt**" button to stop core ID 0
- **Right click** on the Probe entry for core ID 0 and left click on "**Open in New Window**" in the popup window



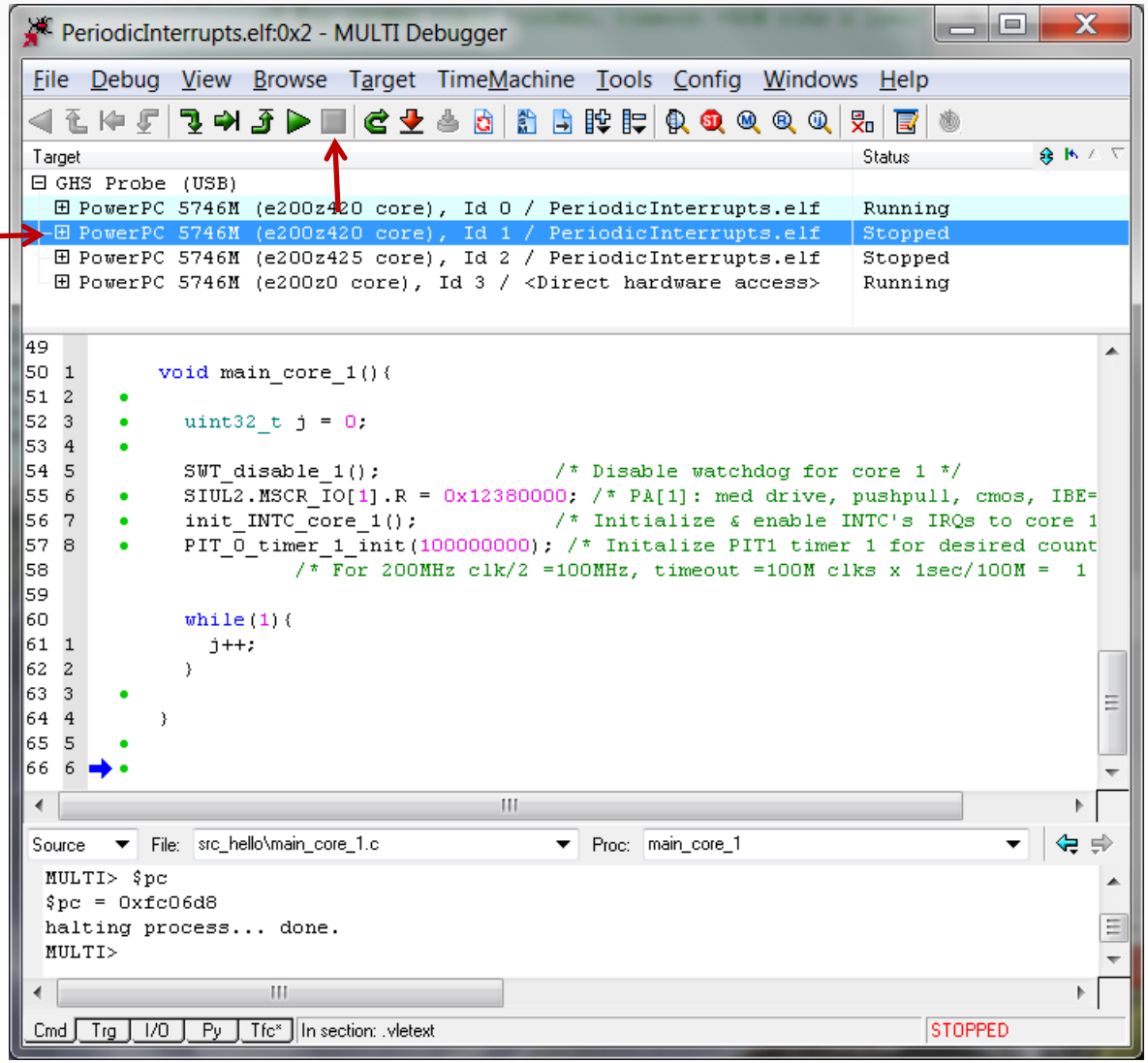
Hands-on 4.22: **Multicore Debugging 2**

- A colored, dedicated debugging window for core ID 0 appears
- Enter "**update 1**" in the command window
- **Left Click** on the "**Go**" button
- **Double click** on the variable **i** to put it in a Data Explorer window
- The variable display will update every second



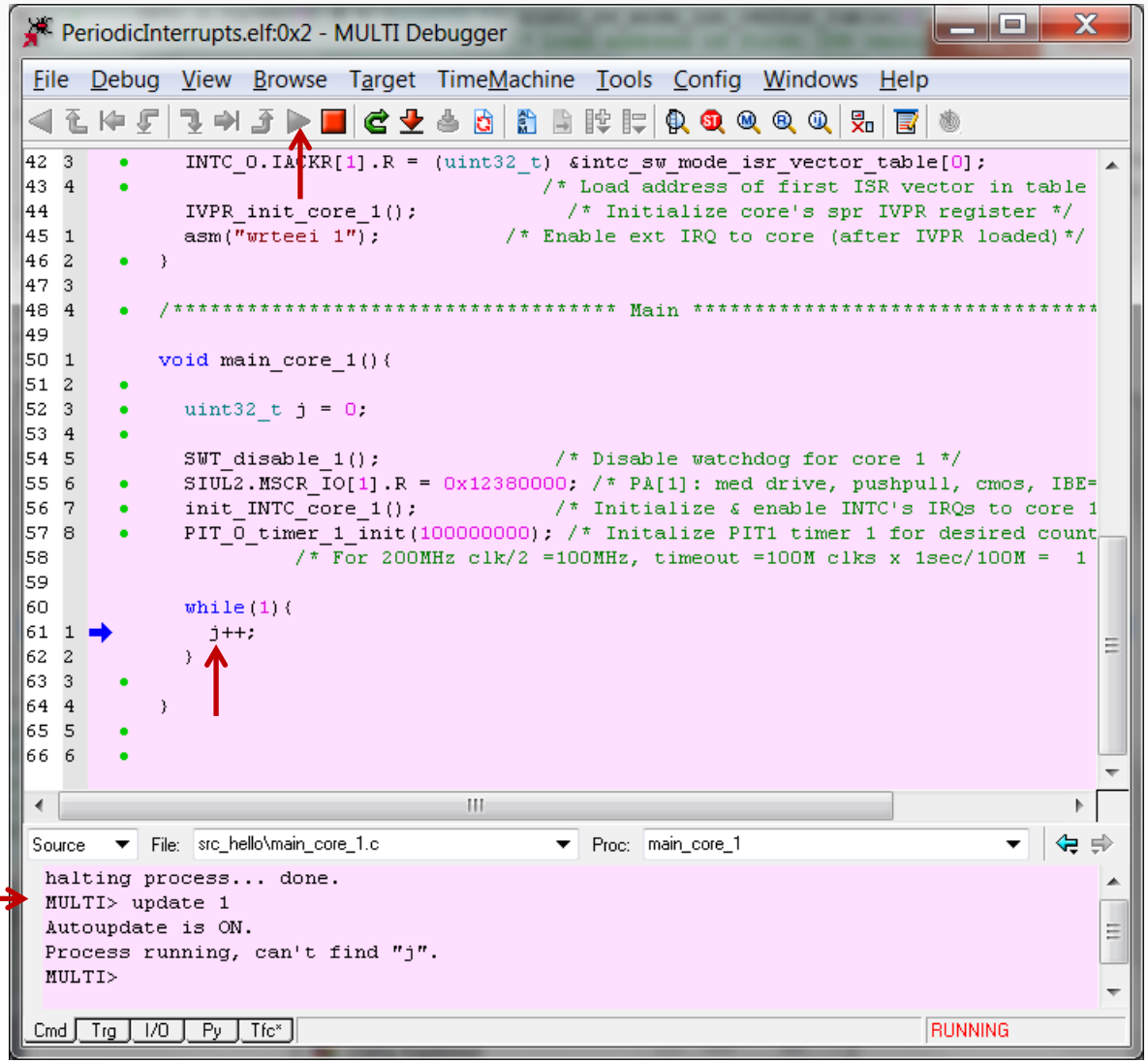
Hands-on 4.22: **Multicore Debugging 3**

- **Left click** on the Probe entry for core ID 1 and you will see that it is running
- **Left Click** on the "**Halt**" button to stop core ID 1
- **Right click** on the Probe entry for core Id 1 and left click on "**Open in New Window"** in the popup window



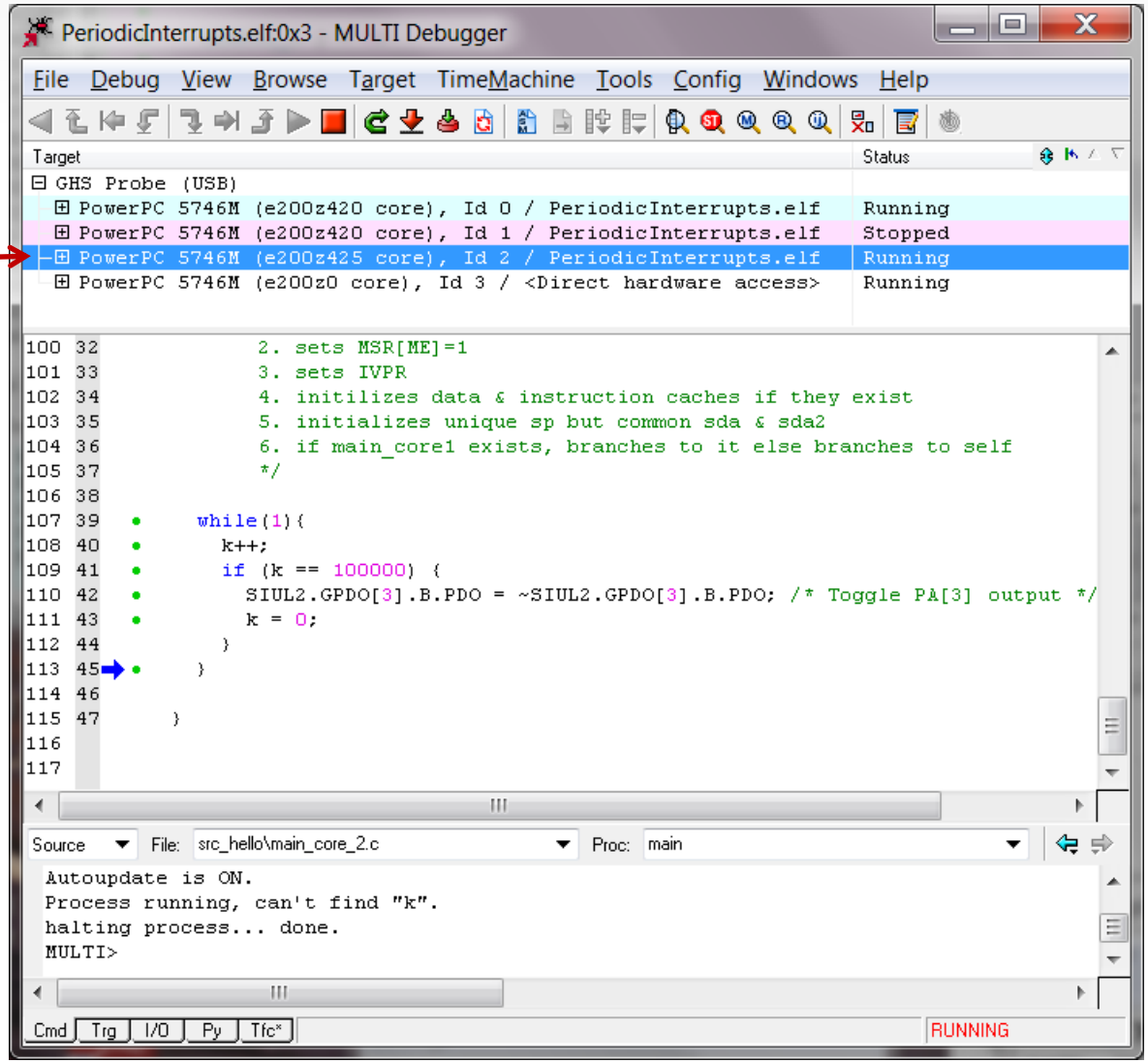
Hands-on 4.22: **Multicore Debugging 4**

- A colored, dedicated debugging window for core ID 1 appears
- Enter "**update 1**" in the command window
- **Left Click** on the "**Go**" button
- **Double click** on the variable **j** to put it in a Data Explorer window
- The variable display will update every second



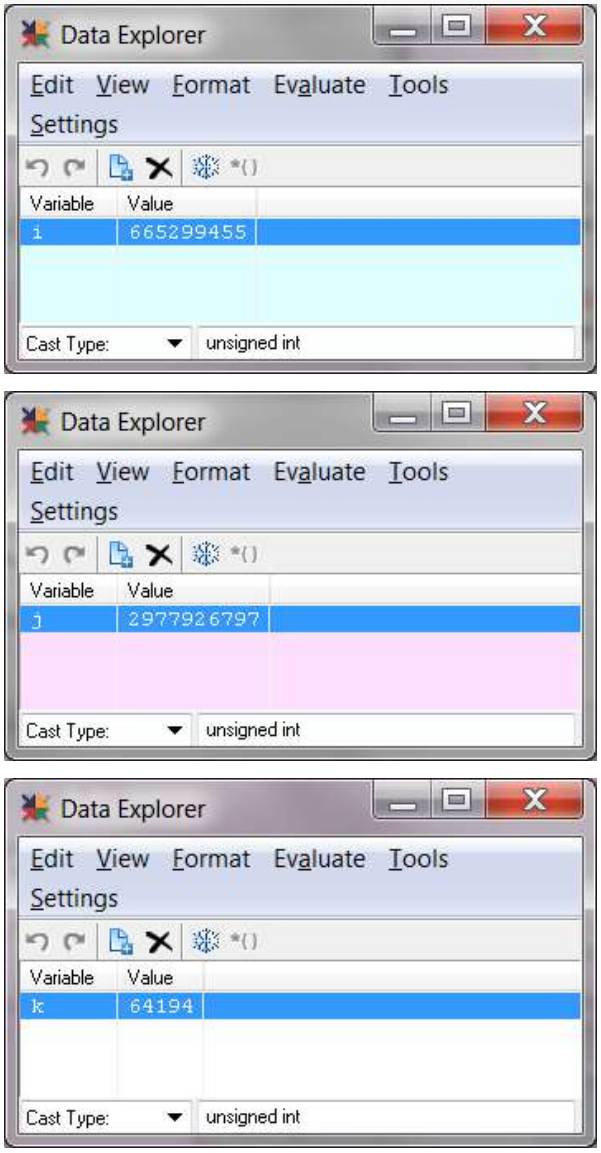
Hands-on 4.22: **Multicore Debugging 5**

- **Left click** on the Probe entry for core ID 2
- Enter "**update 1**" in the command window
- **Left Click** on the "**Go**" button
- **Double click** on the variable **k** to put it in a Data Explorer window
- The variable display will update every second
- You can either debug in the white debug window or open a colored, dedicated debug window.



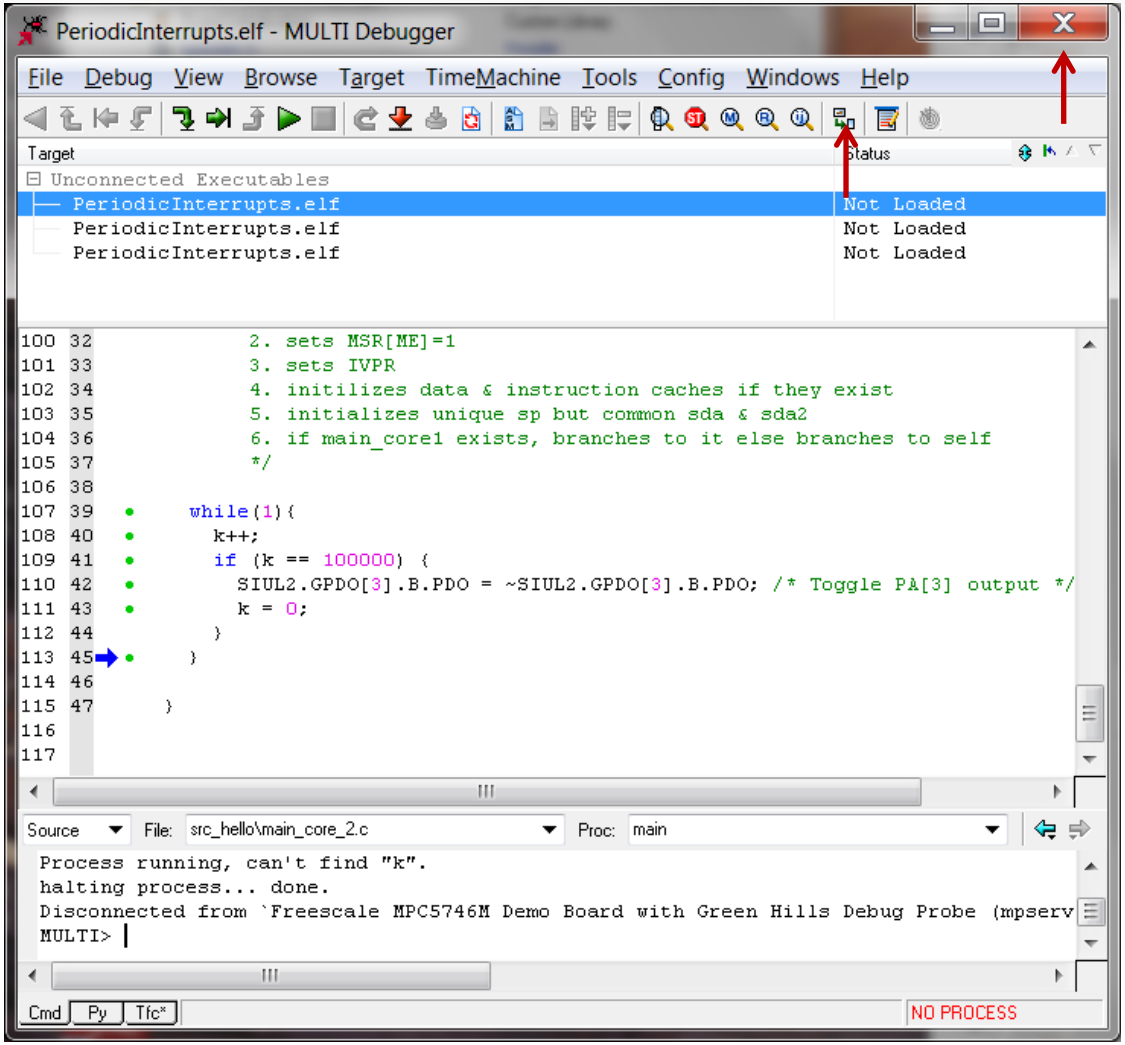
Hands-on 4.22: **Multicore Debugging 6**

- All windows derived from a debugging window have the same coloring as the debugging window



Hands-on 4.22: **Multicore Debugging 7**

- **Left click** on the "**Disconnect**" button in the white Debugger window to close the debugging connection
- **Close** the white debugging window
- Note that all of the other debugging windows close as well

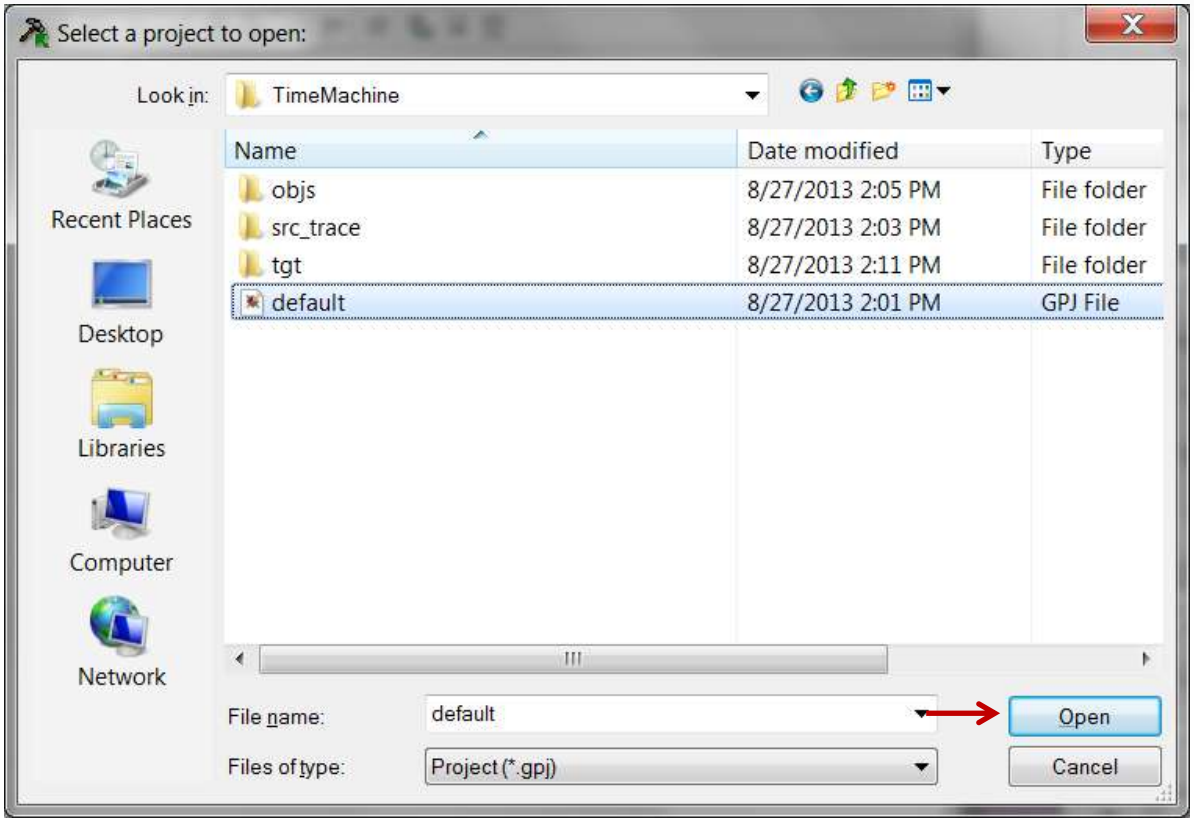


Hands-on 5:

- Advanced Tools
  1. TimeMachine Debugger
  2. Runtime error check
  3. DoubleCheck
  4. MISRA checking

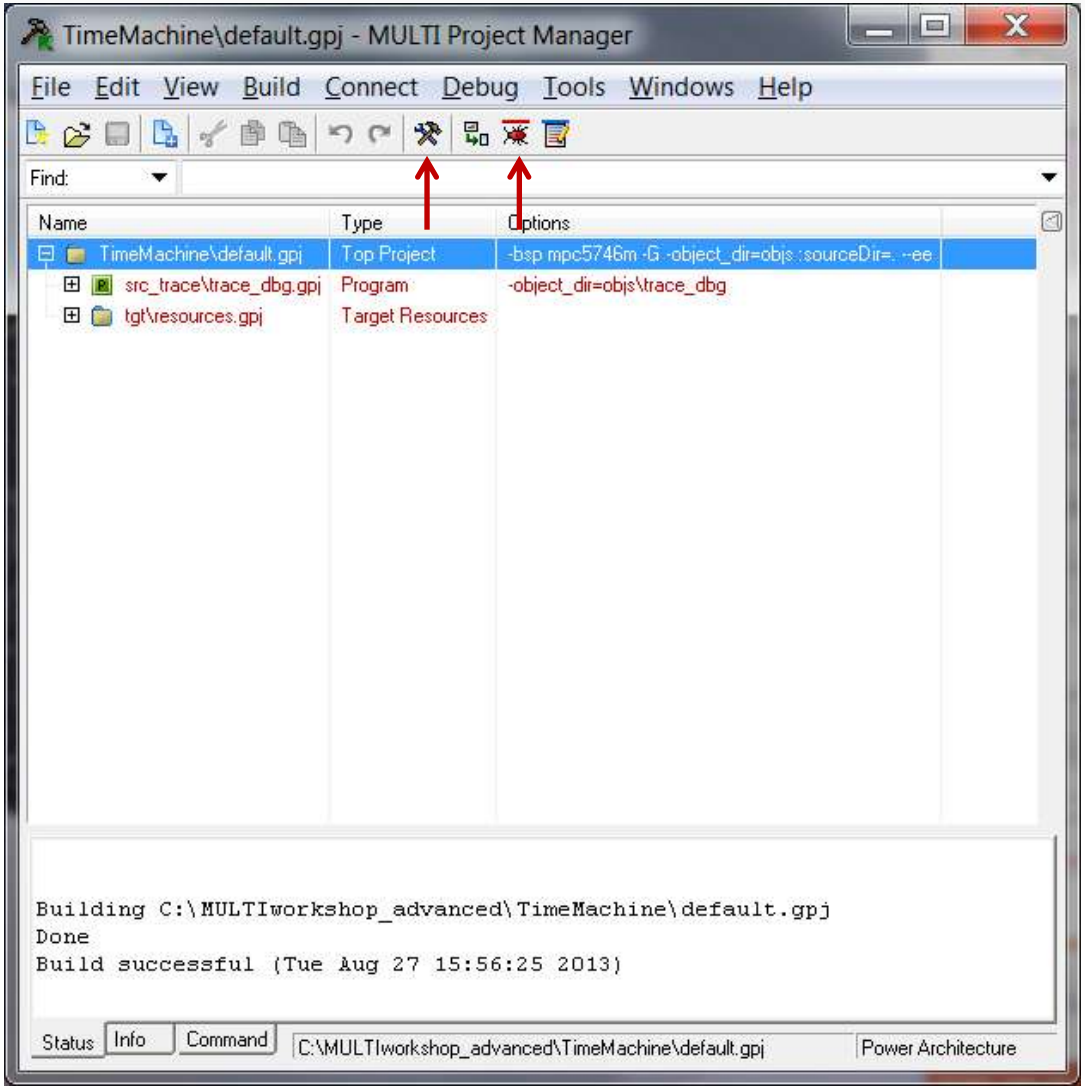
Hands-on 5.1.1: **Load the TimeMachine Project**

- **Left click** on **File-> Open Project** in the Project Manager window
- **Navigate** to **MULTIworkshop_advanced\TimeMachine\default.gpj**
- Left click on the "**Open**" button



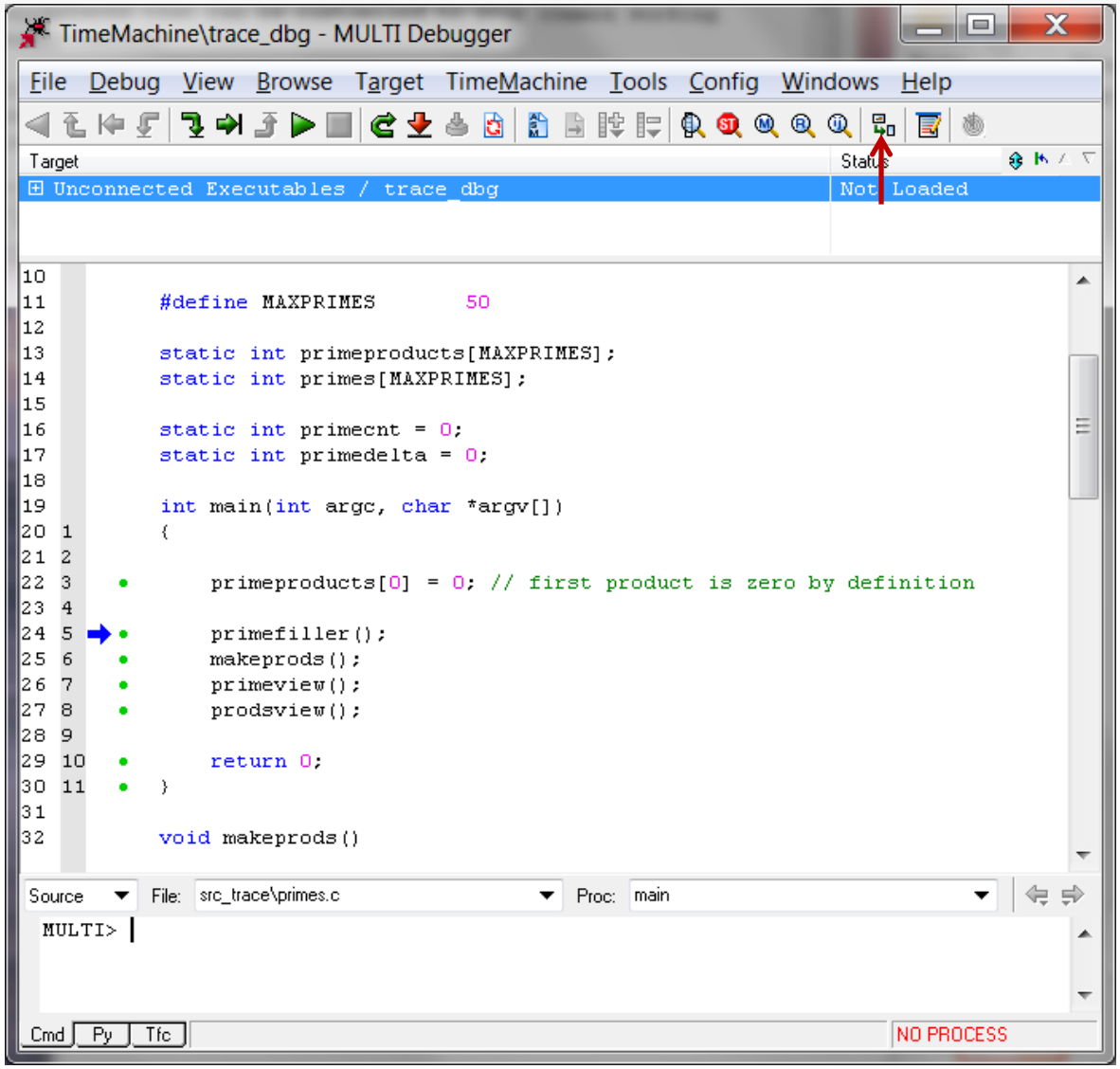
Hands-on 5.1.2: **Build the TimeMachine Project**

- **Left click** the "**Build**" button in the Project Manager window
- **Left click** the "**Debug**" button in the Project Manager window



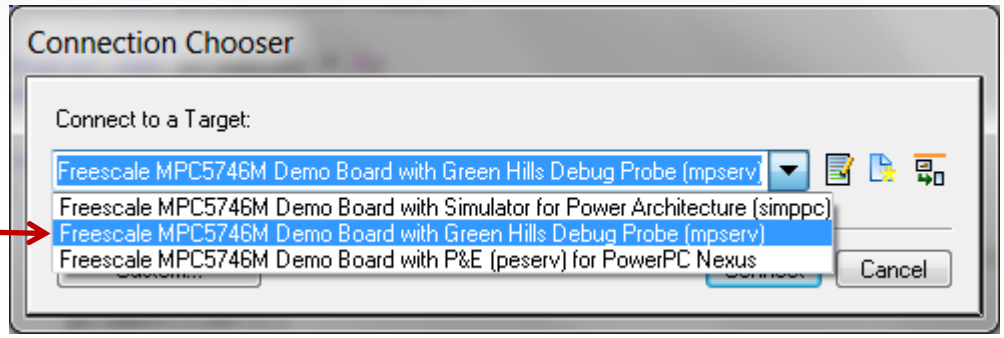
Hands-on 5.1.3: **Start the Debugger - 1**

- **Left click** the "**Connect**" button in the Debugger window

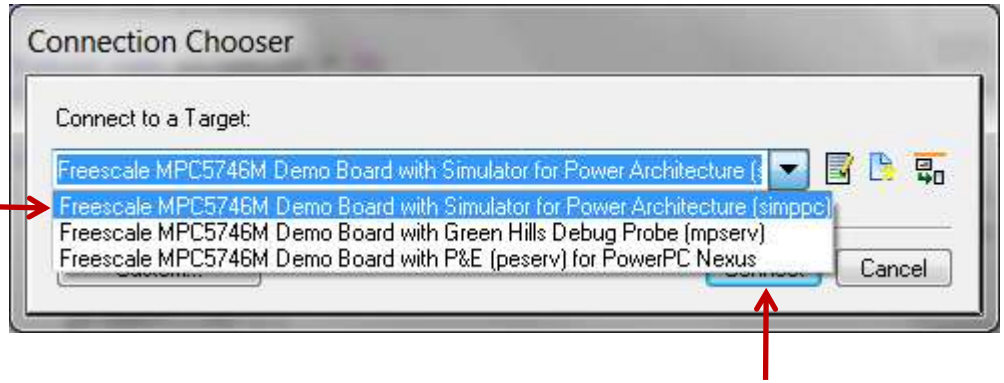


Hands-on 5.1.3: **Start the Debugger - 2**

- If you have a Supertrace probe, select **Green Hills Probe** in the pull down menu

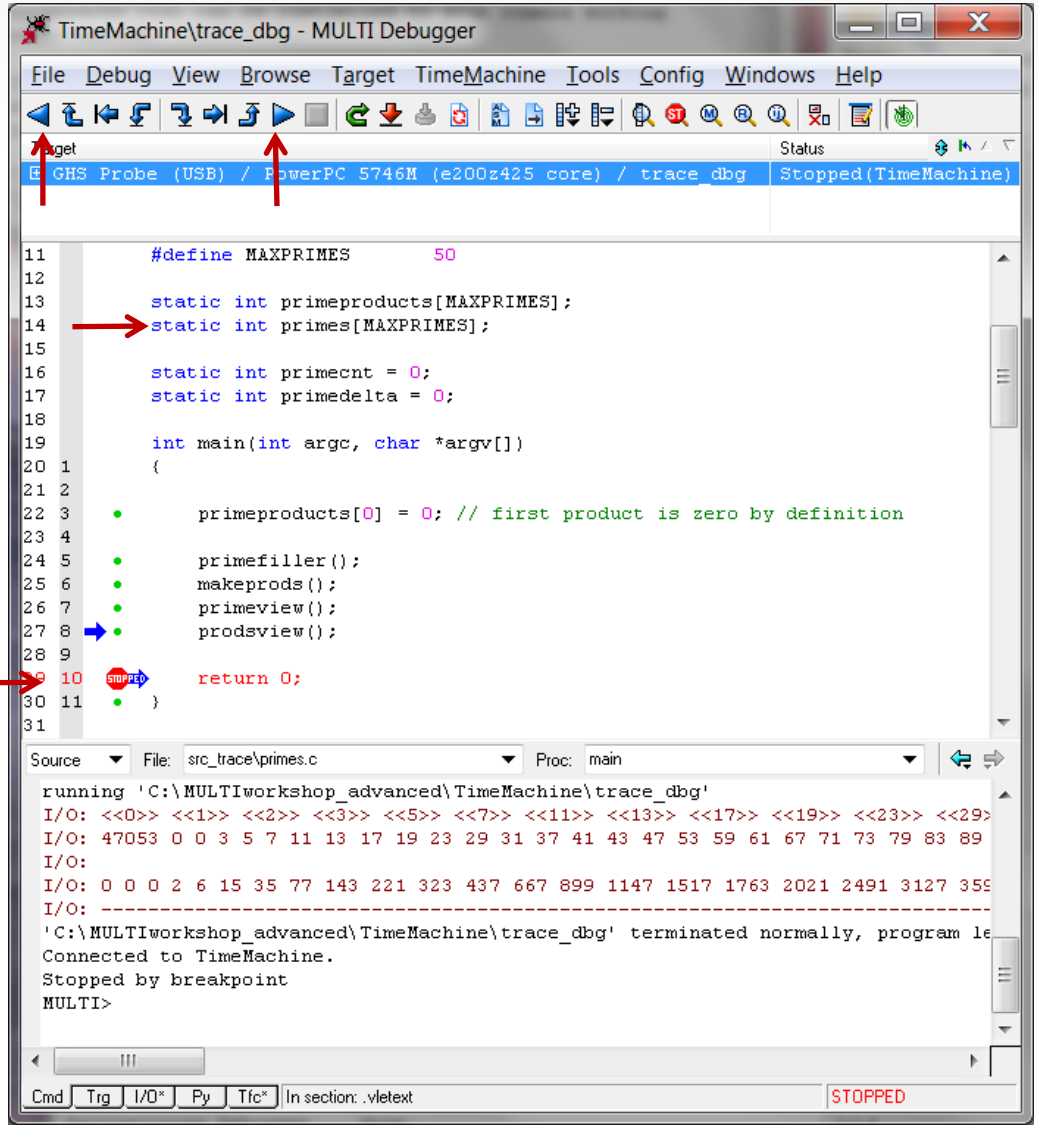


- If you have a Green Hills probe, select **Simulator** in the pull down menu
- **Left click** the "**Connect**" button



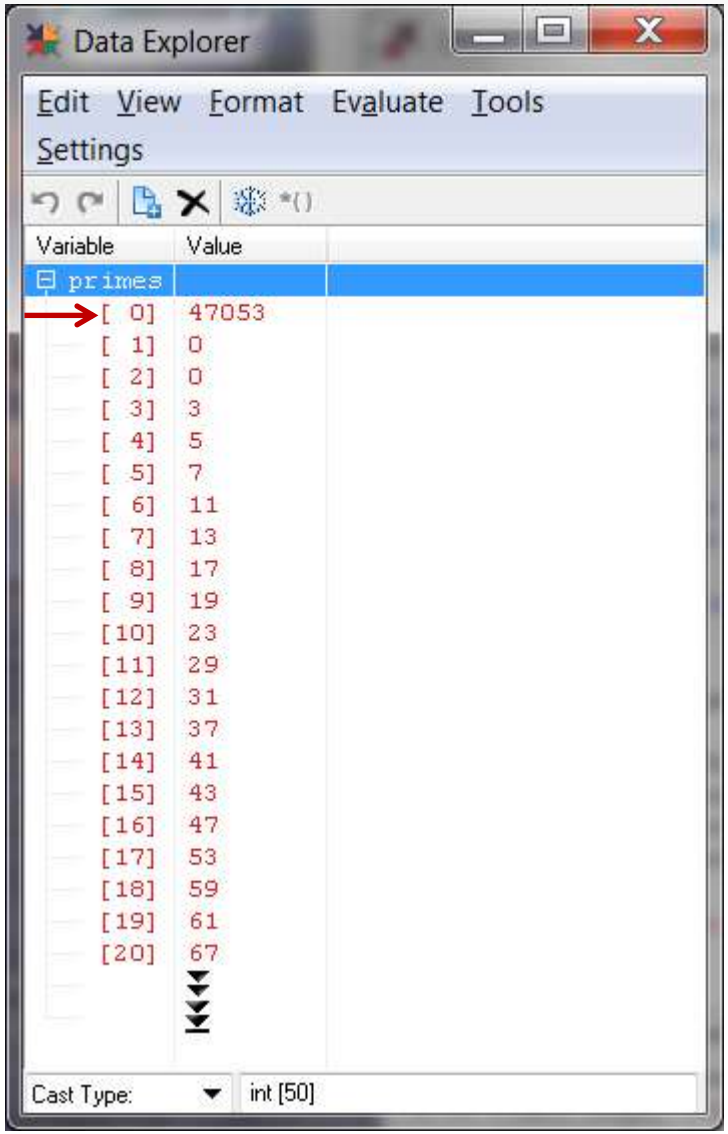
Hands-on 5.1.4: **Run the Program**

- **Left click** the “**Go**” button in the Debugger window
- Notice program “terminated normally” per status window message
- Set a **breakpoint** on Main:10
- **Left click** the “**Go Back**” (top left) button to start TimeMachine
- **Double click** on primes in at primes.c:14 to view primes in a data explorer window



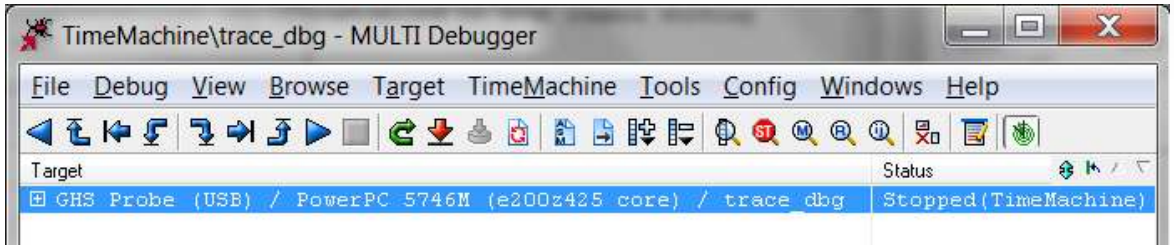
Hands-on 5.1.5: **Examine the Results**

- Note that the list of prime numbers is incorrect
- primes[0] is wrong

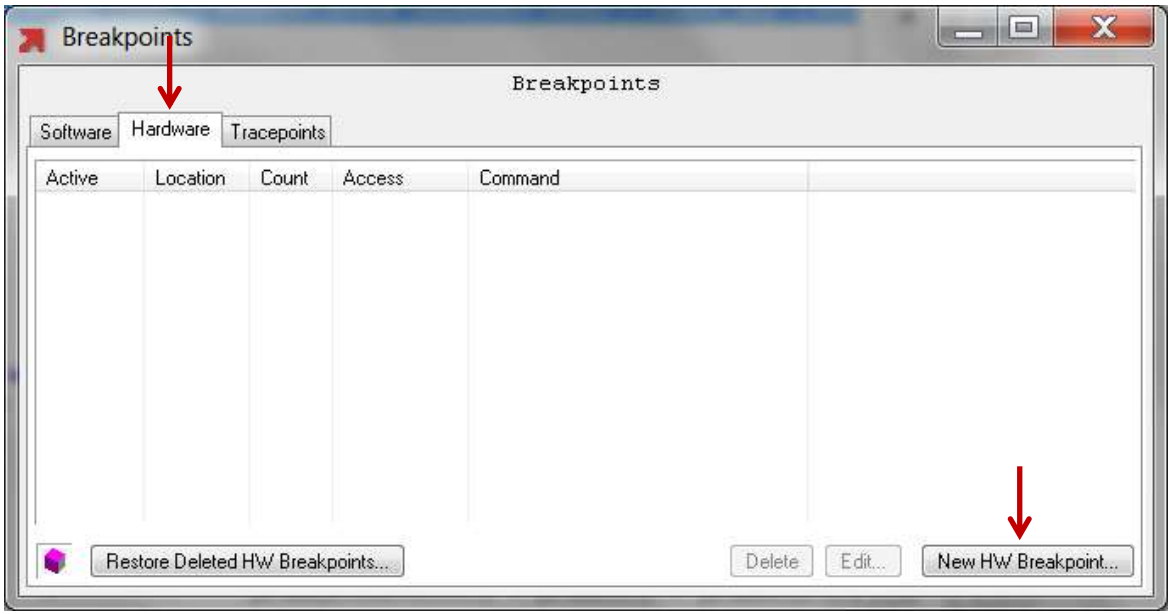


Hands-on 5.1.6: **Set a Data Breakpoint - 1**

- **Left click** the "**Breakpoints**" button in the Debugger window

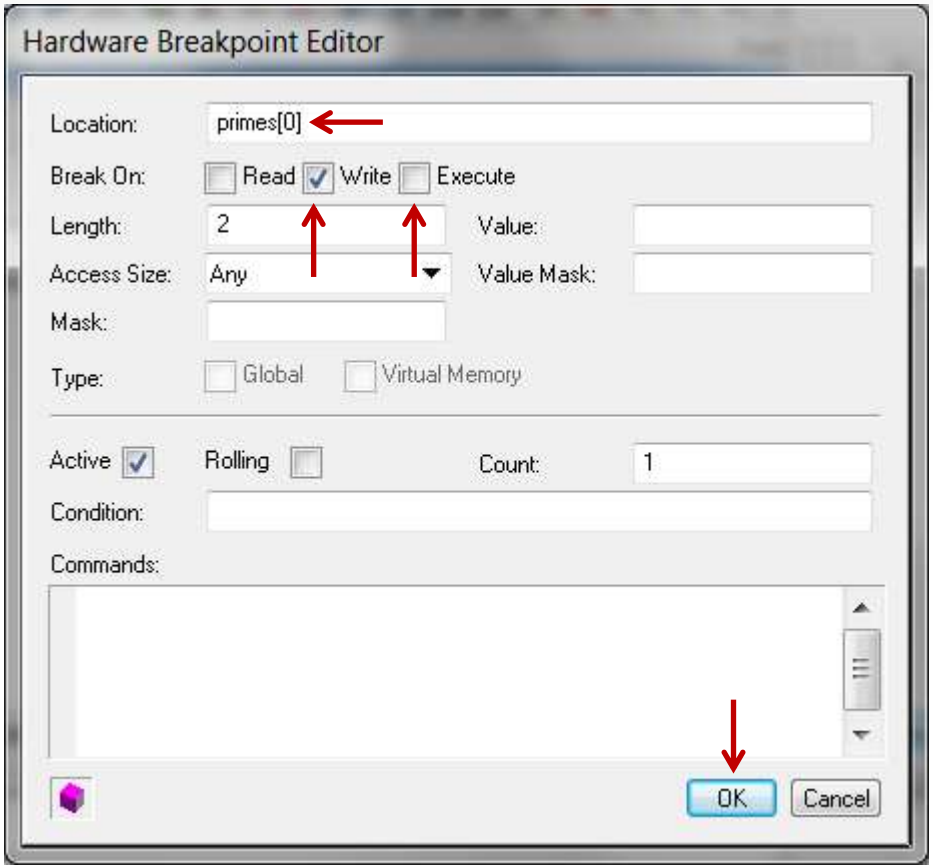


- **Left click** on the "**Hardware**" tab
- **Left click** on the "**New HW Breakpoint**" button



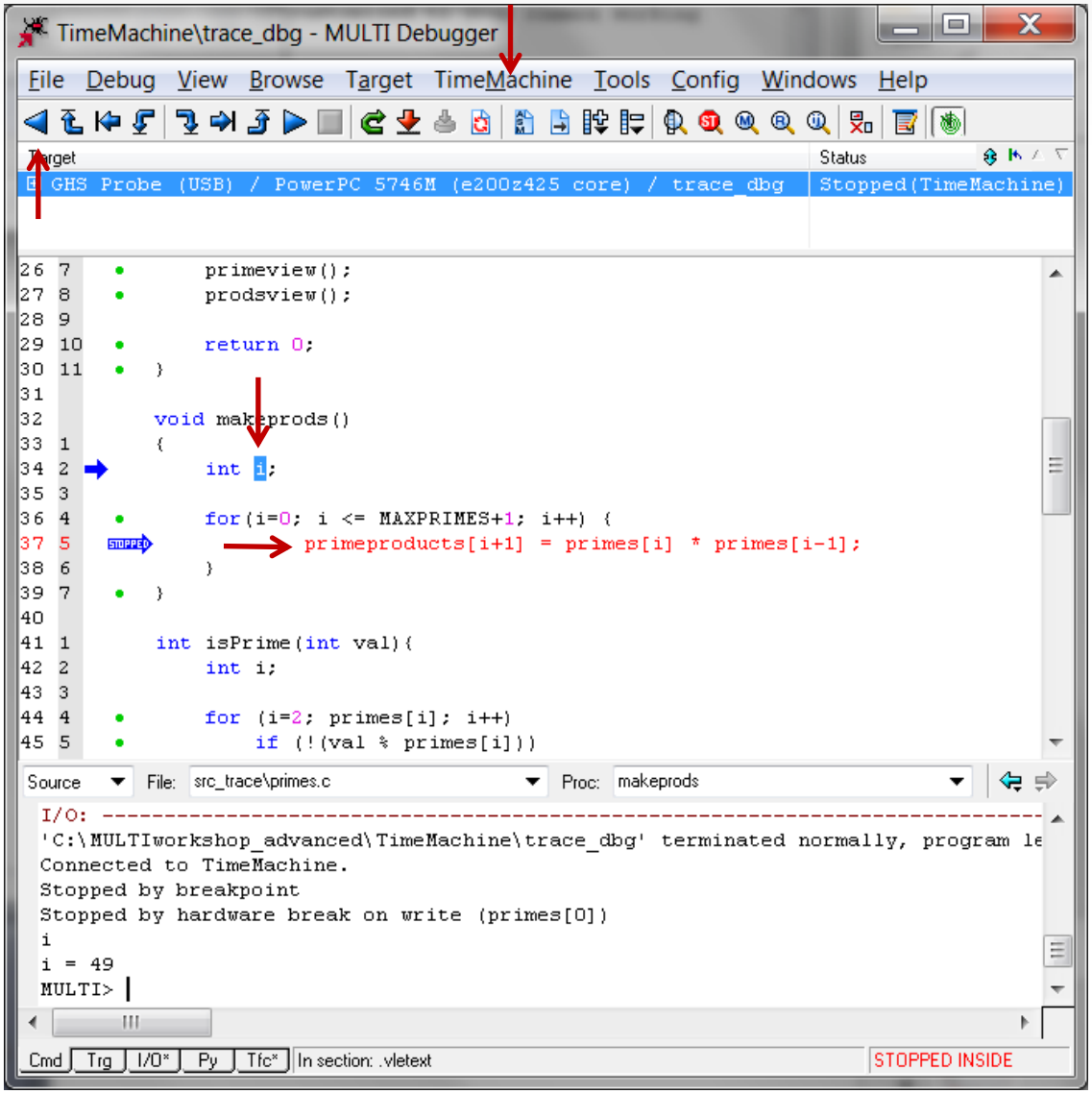
Hands-on 5.1.6: **Set a Data Breakpoint - 2**

- Enter **PS.primes[0]** in the Hardware Breakpoint Editor window
- **Left click** to select **break on write**
- **Left click** to deselect **break on Execute**
- Left click the "**OK**" button
- **Close** the Breakpoints window



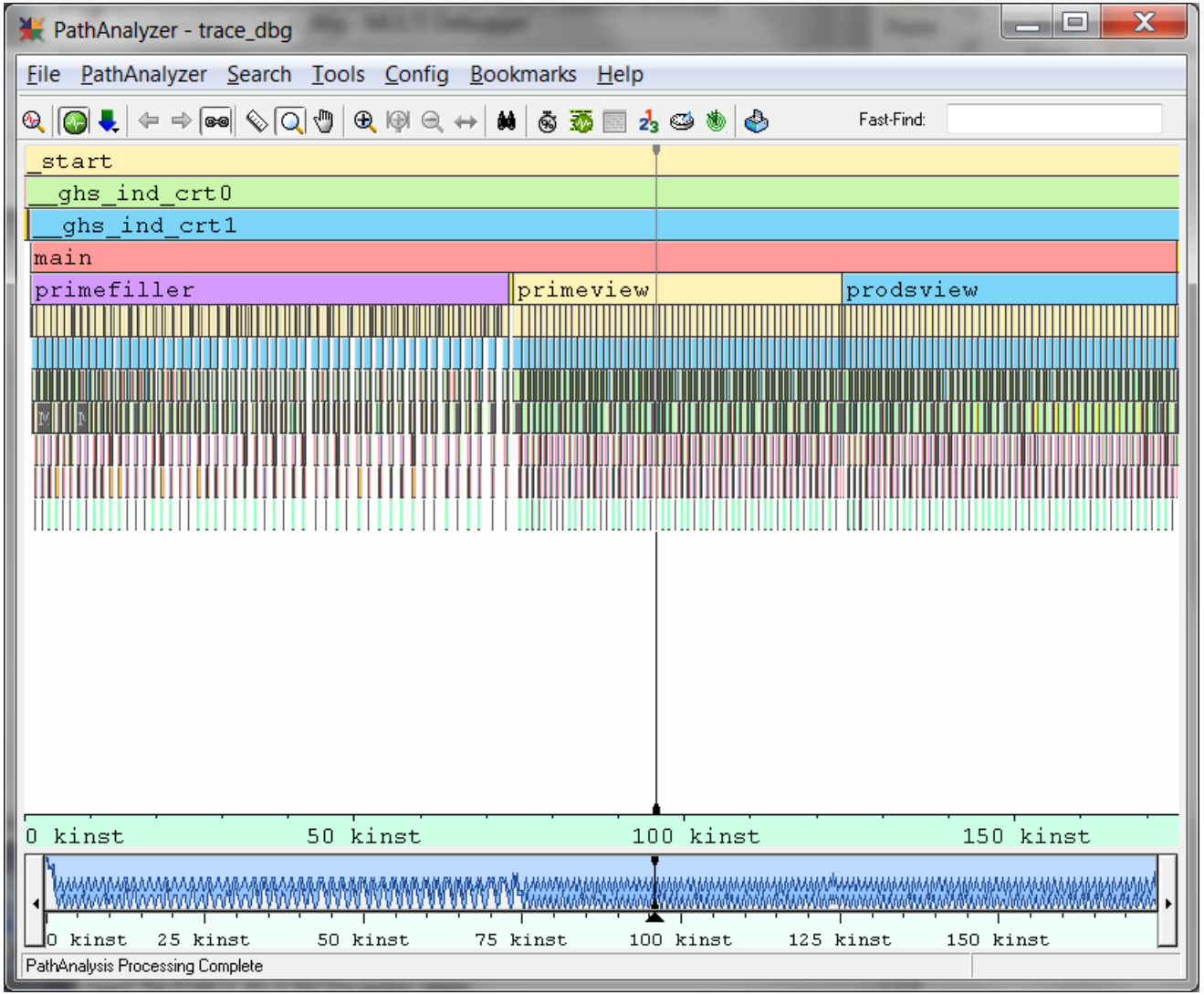
Hands-on 5.1.7: **Run Back to the Breakpoint**

- **Left click** the "**Go Back**" button in the Debugger window
- Note that primes[0] is being modified by a write to primeprods()
- **Left click** the variable **i** in line 2 of function makeprods
- Note I = 49 in status window
- In **Time Machine** menu of the debugger window, left click on **Path Analyzer**



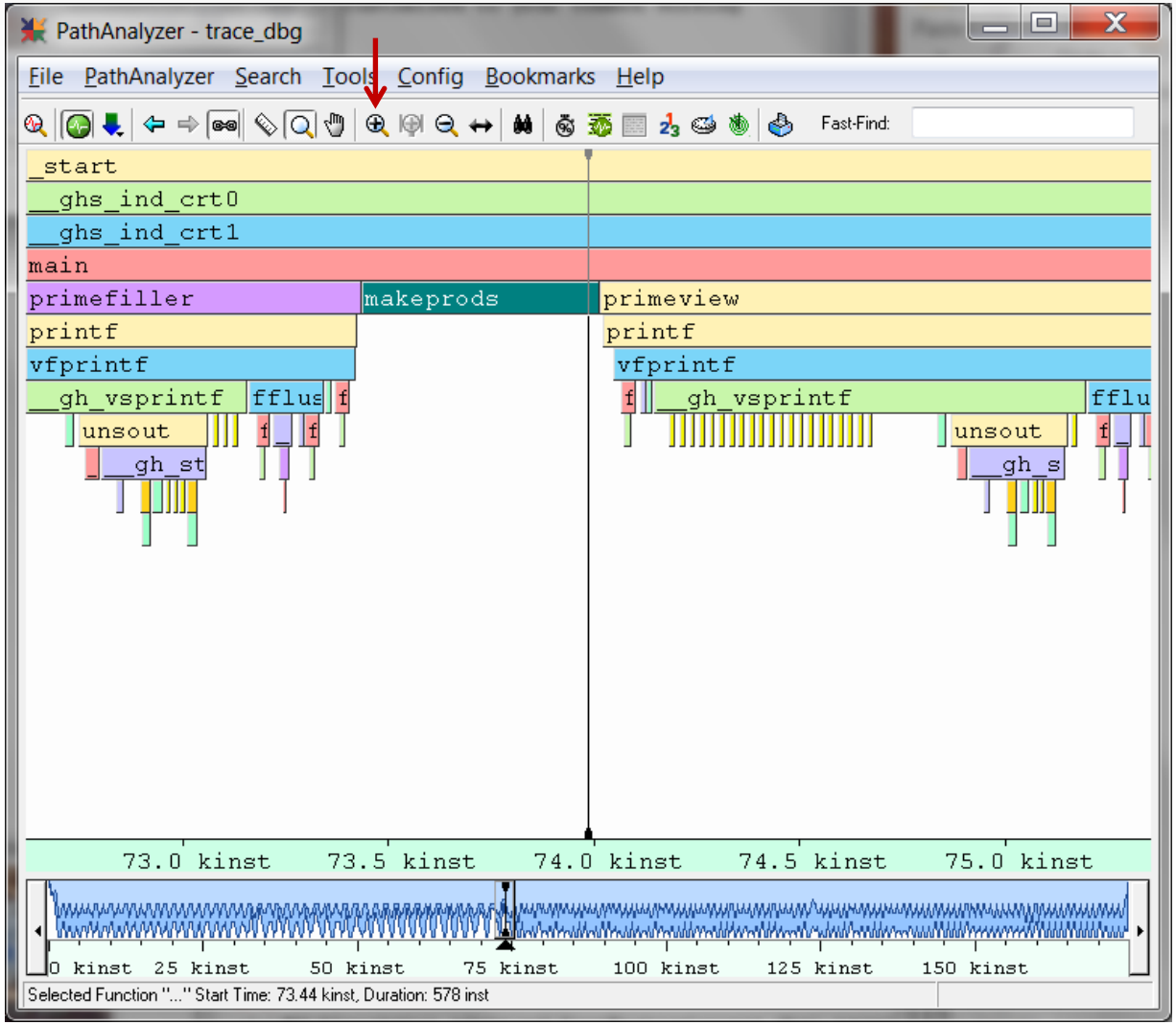
Hands-on 5.1.8: **The Path Analyzer - 1**

- A call-level view of the entire run is displayed



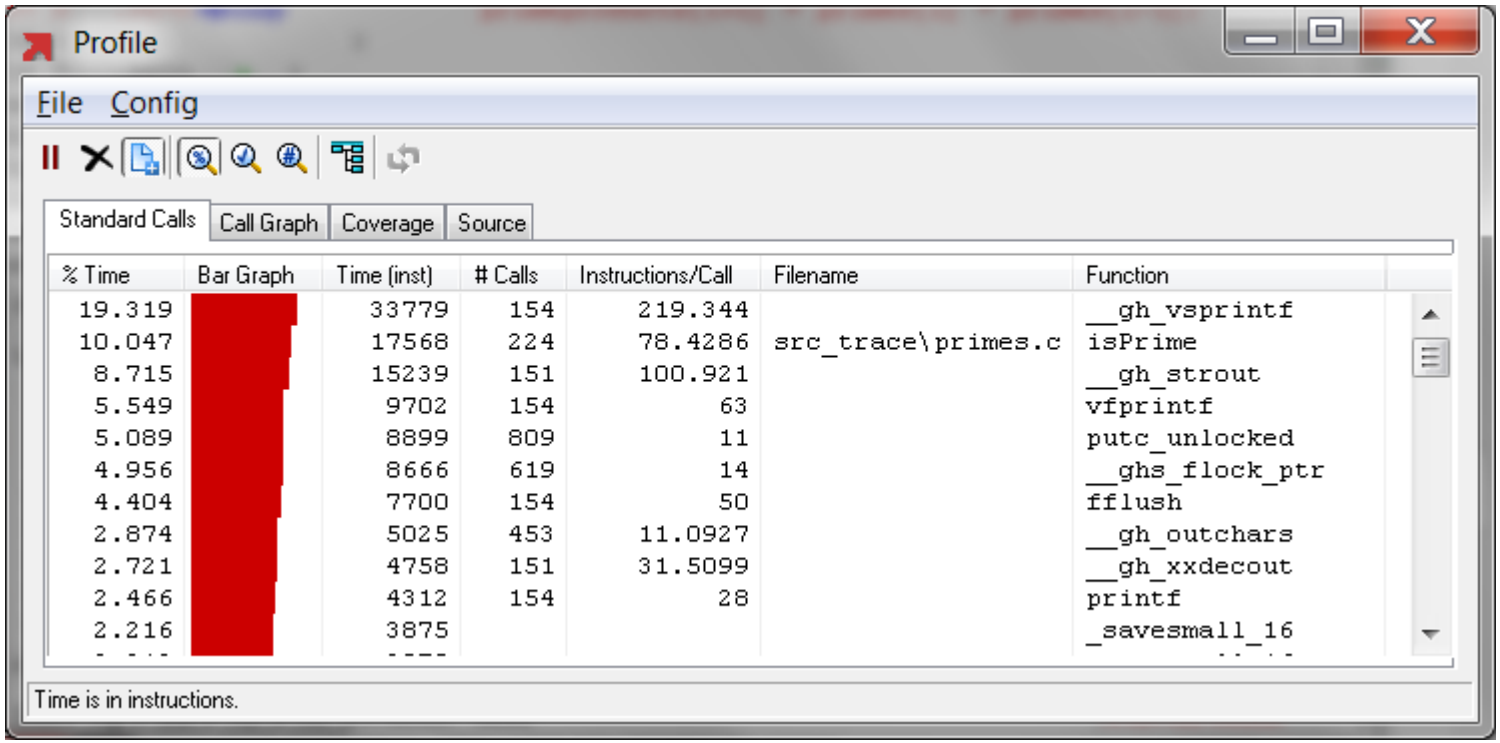
Hands-on 5.1.8: **The Path Analyzer - 2**

- **Left click** a few times on the "**zoom-in**" button in the **Path Analyzer** window
- More detail becomes visible
- Search and breakpoints are used to locate interesting activity



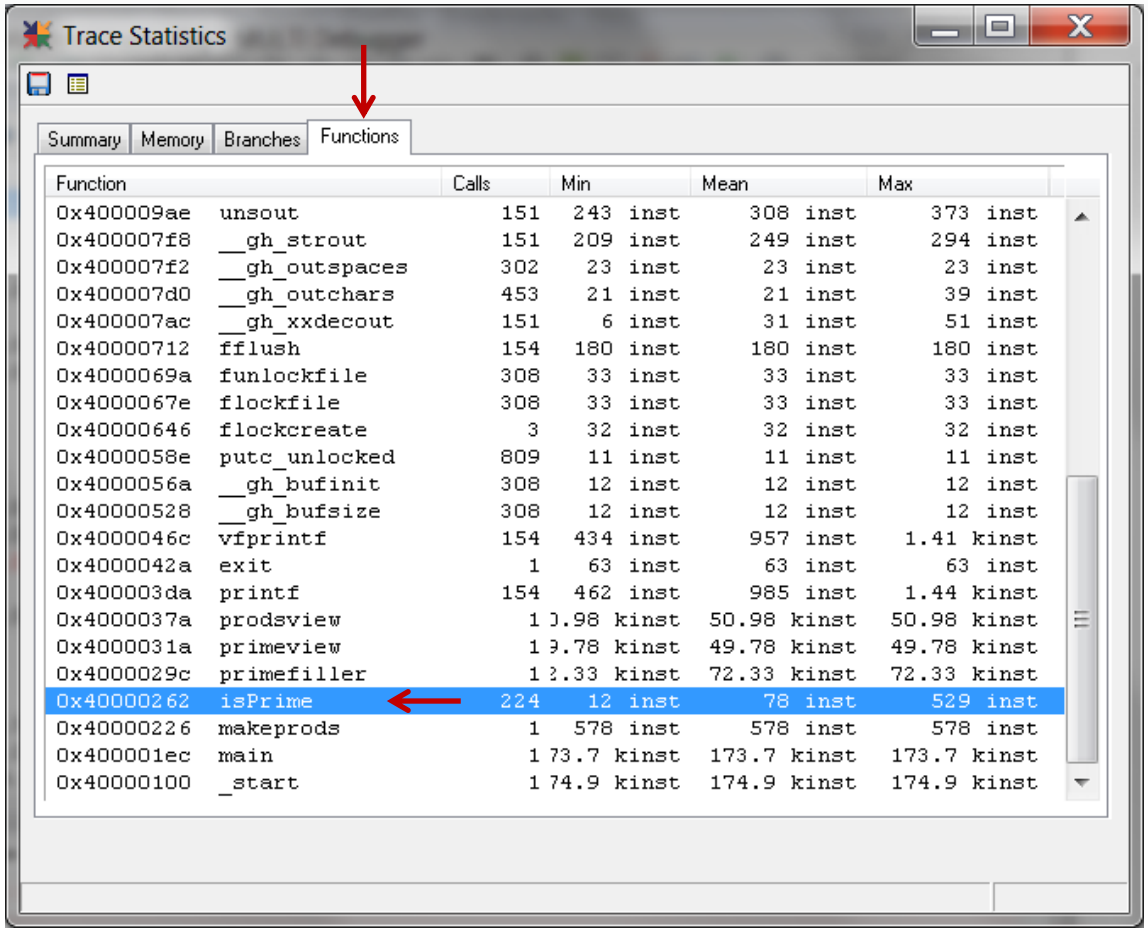
Hands-on 5.1.9: **Profiling**

- In **Time Machine** menu of the debugger window, **left click** on **Profile**.
- A bar graph of cumulative function execution time is displayed



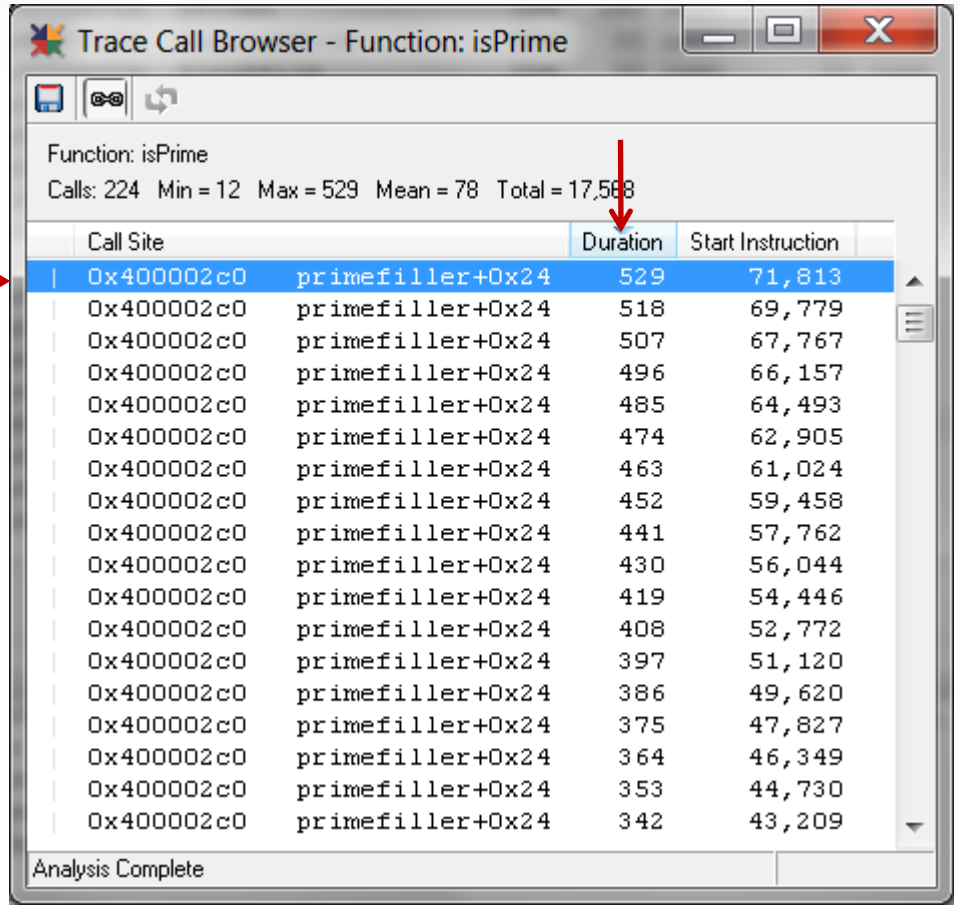
Hands-on 5.1.10: **Trace Statistics - 1**

- In **Time Machine** menu of the debugger window, left click on **Trace Statistics**
- **Left click** on the **Functions** tab
- **Double click** on **isPrime**



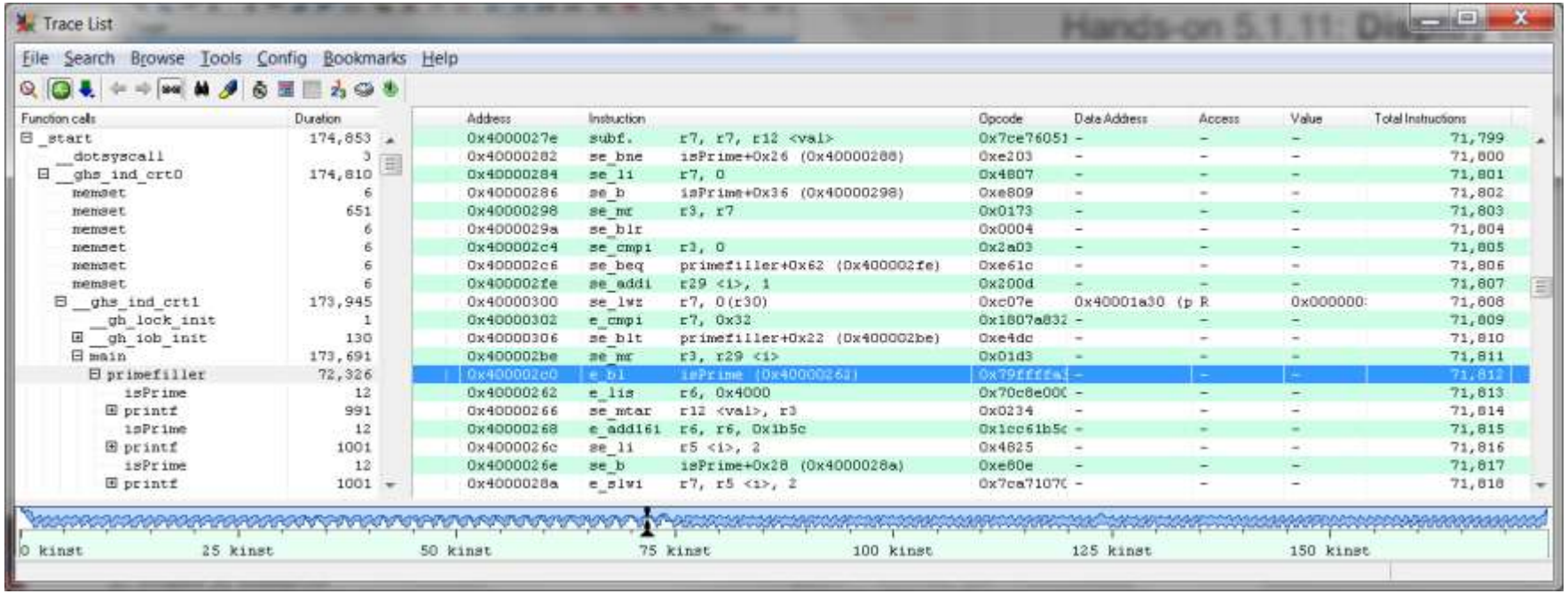
Hands-on 5.1.10: **Trace Statistics - 2**

- Every instance of isPrime execution is displayed
- **Left click** on “duration” to sort them by execution time.
- **Double click** on the longest **instance** of isPrime
- The Debugger and Path Analyzer windows display that instance



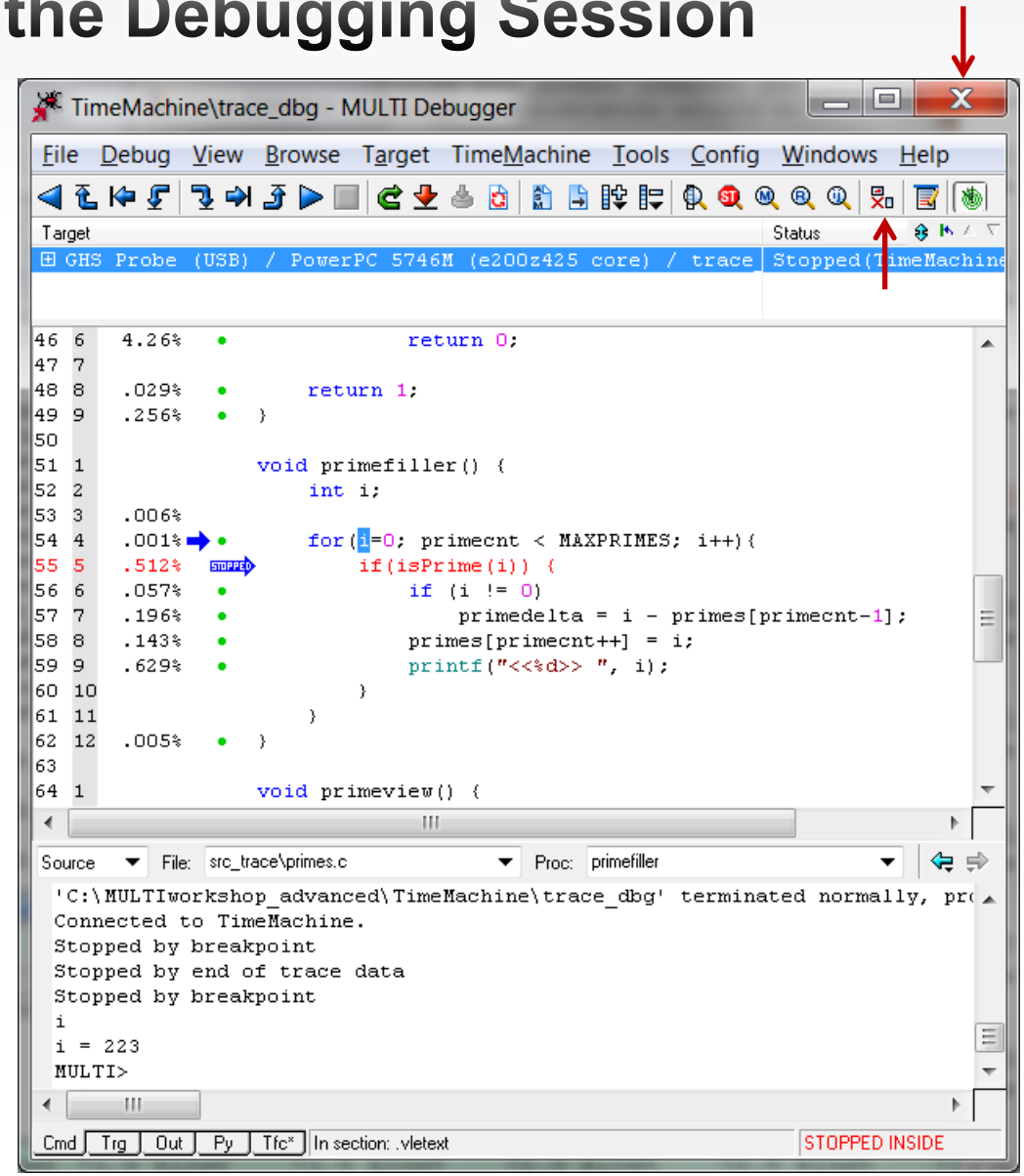
Hands-on 5.1.11: **Display the Trace List**

- In **Time Machine** menu of the debugger window, left click on **Trace List**
- Note the function-level and assembly-level trace



Hands-on 5.1.12: **End the Debugging Session**

- **Left click** on the "**Disconnect**" button to close the debugging connection
- **Close** the Debugger window
- Note that all of the other windows opened from the Debugger close as well

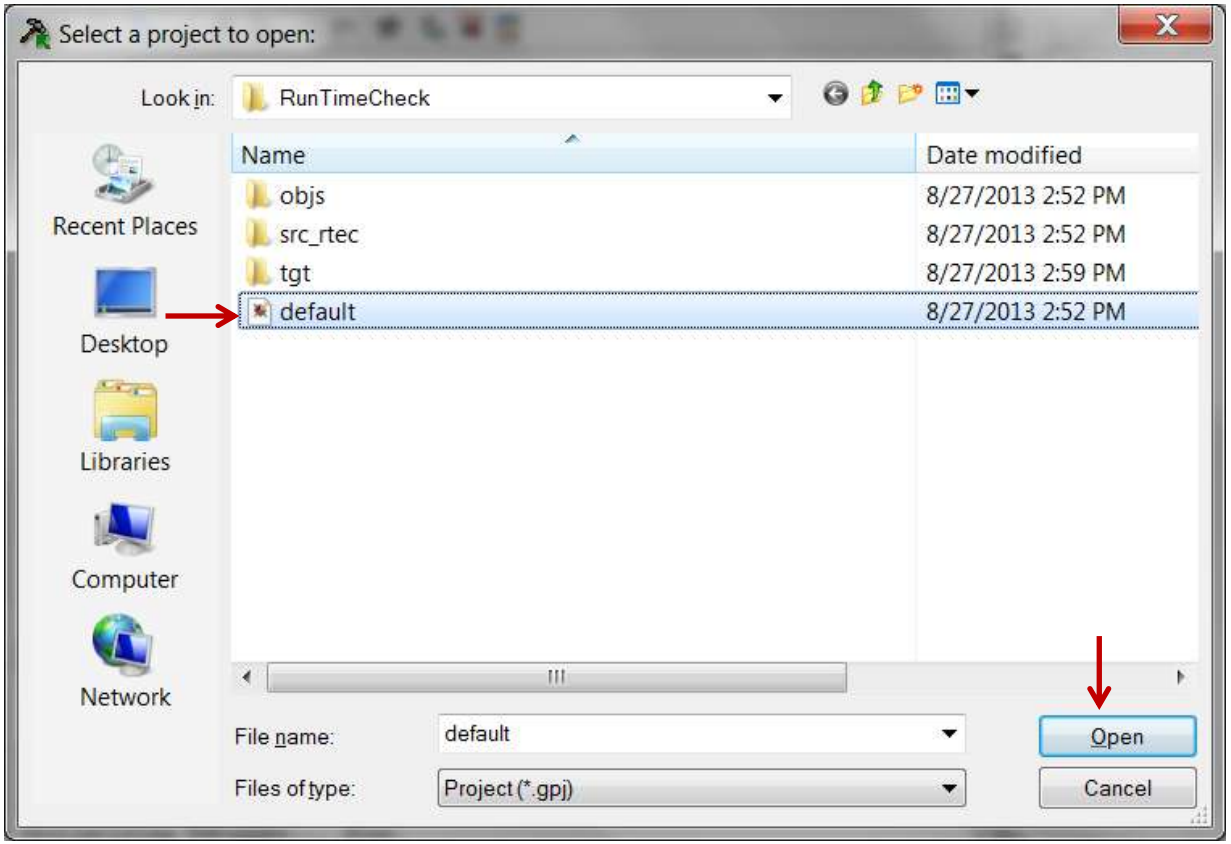


Hands-on 5:

- Advanced Tools
  1. TimeMachine Debugger
  2. Runtime error check
  3. DoubleCheck
  4. MISRA checking

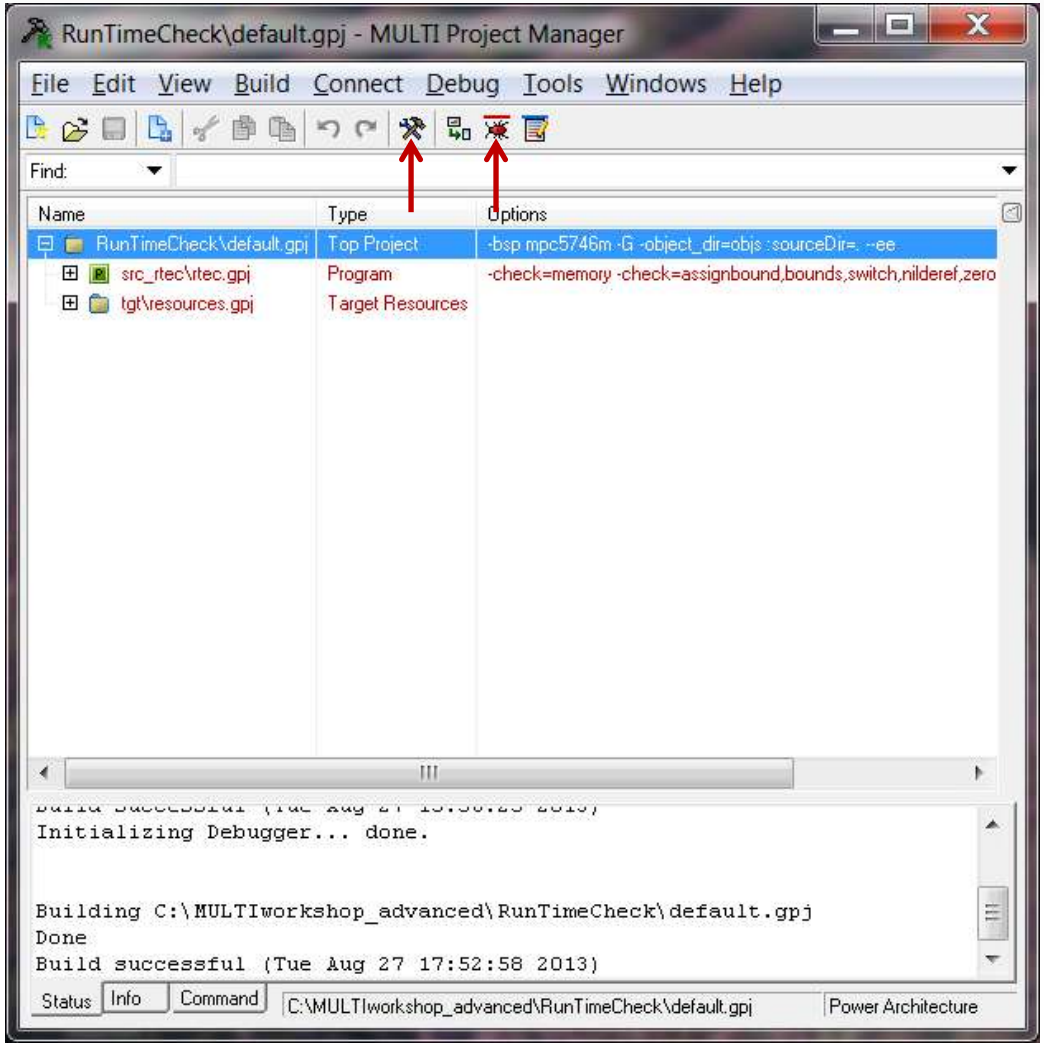
Hands-on 5.2.1: **Load Runtime Error Check Project**

- In the **File** menu of Project Manager, **left click** on **Open Project**
- Navigate to **MULTIworkshop_advanced\RunTimeCheck\default.gpj**
- **Left click** on the "**Open**" button



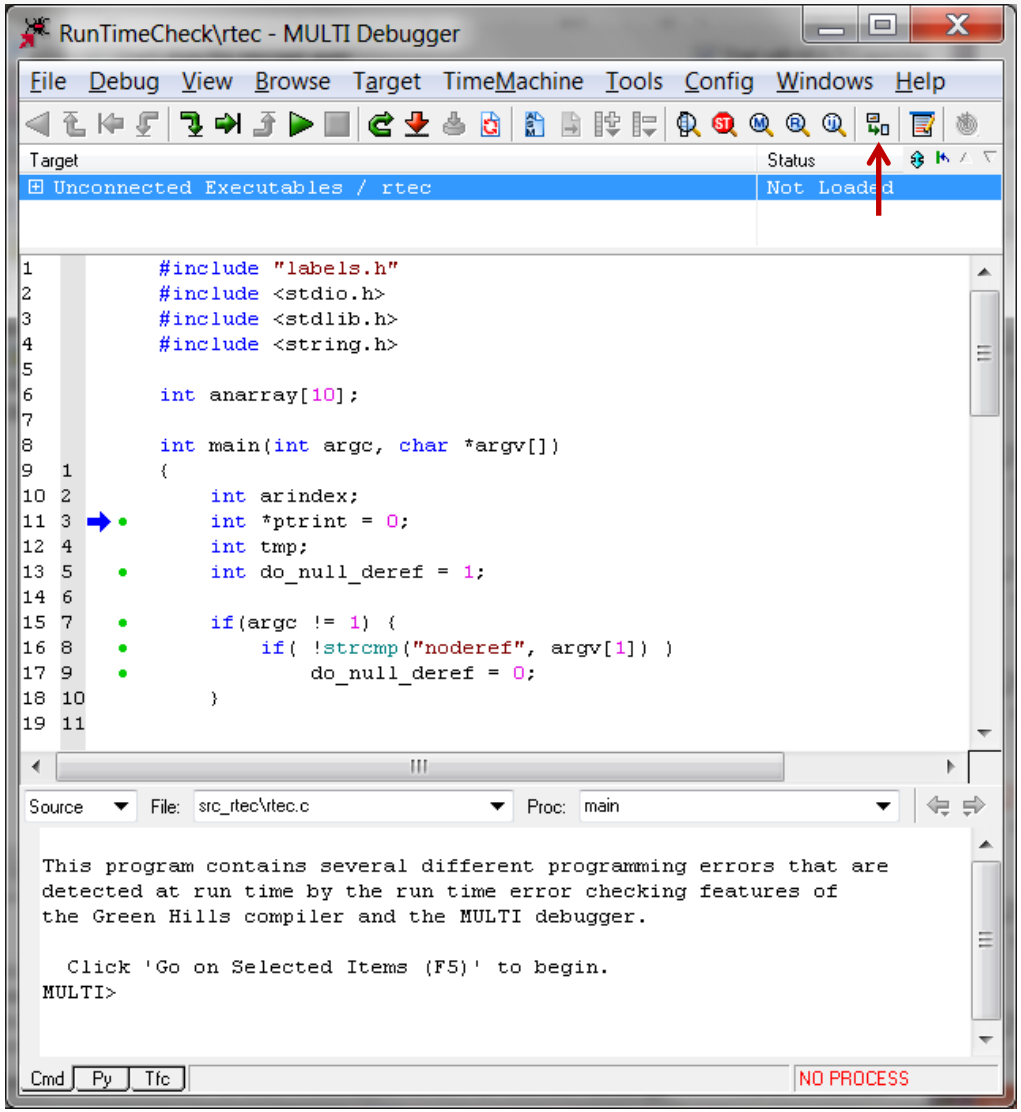
Hands-on 5.2.2: **Build Runtime Error Check Project**

- **Left click** the “**Build**” button in the Project Manager window
- **Left click** the “**Debug**” button in the Project Manager window



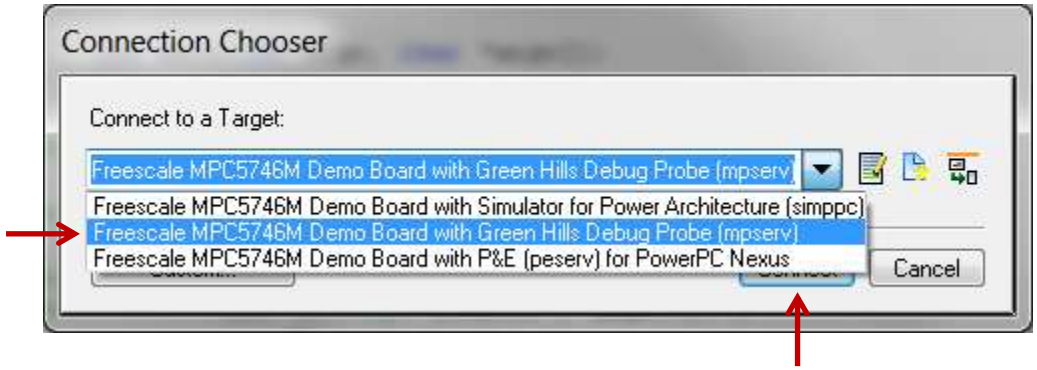
Hands-on 5.2.3: **Start the Debugger - 1**

- **Left click** the "**Connect**" button in the Debugger window



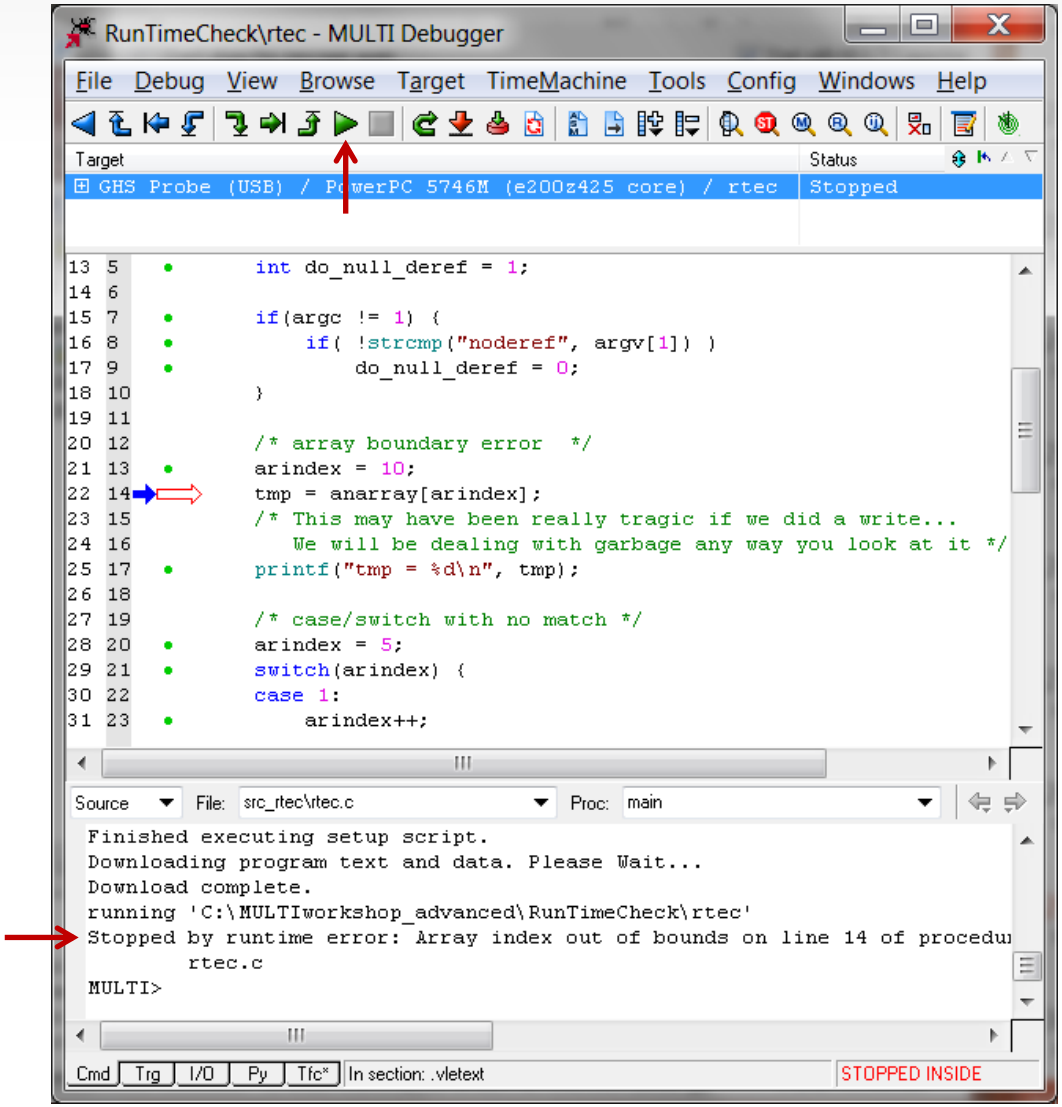
Hands-on 5.2.3: **Start the Debugger - 2**

- **Select** Green Hills Debug Probe in the pull down menu of the Connection Chooser window
- Left click the “Connect” button



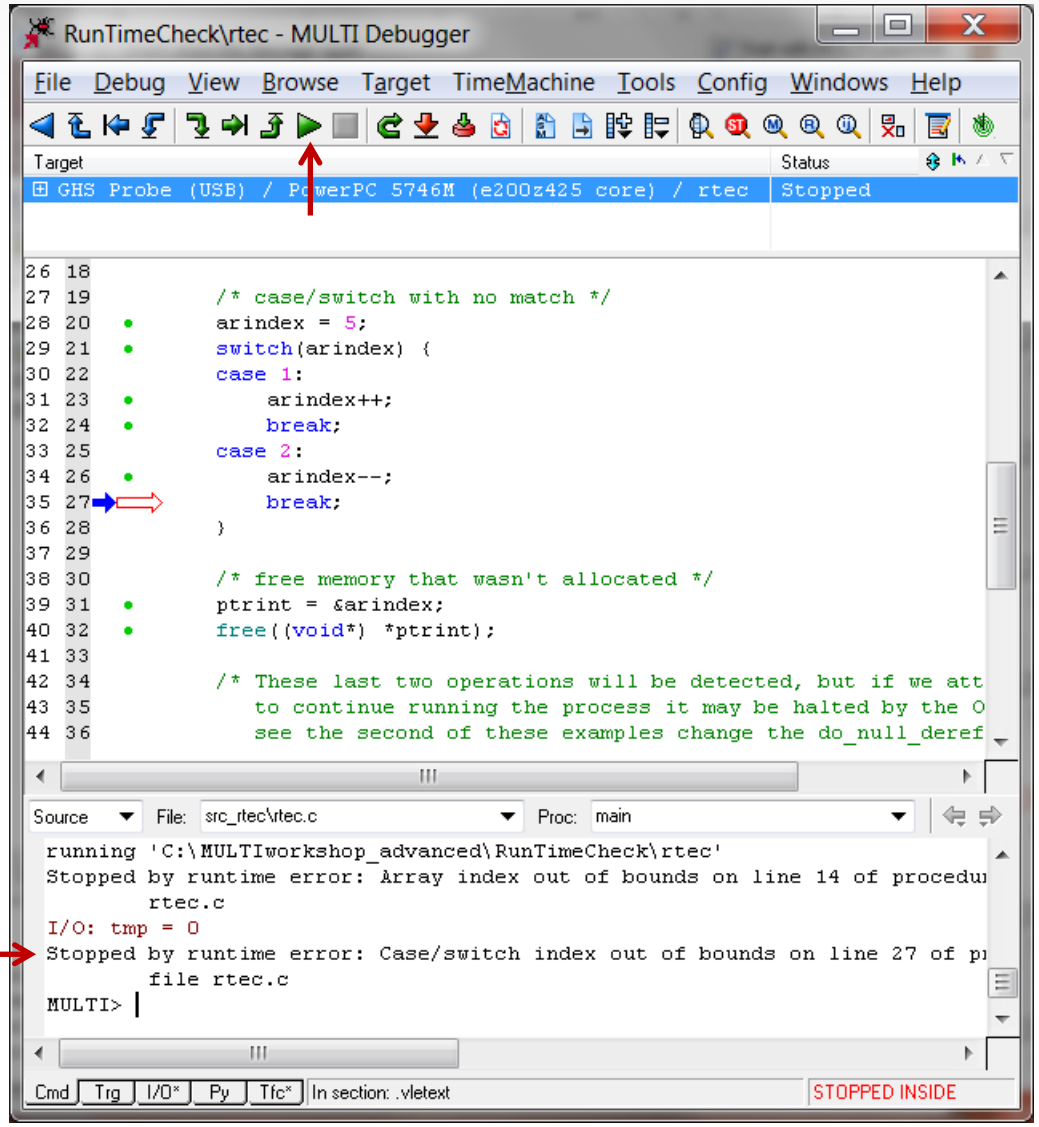
Hands-on 5.2.4: **Run Program and Note First Error**

- **Left click** on the “**Go**” button in the Debugger window
- If the **Prepare Target** window appears, select “Download to RAM” & click OK
- The Initializing Target window will appear briefly.
- Note that the program stops at Main:14 and reports an error
- Left click on the “**Go**” button



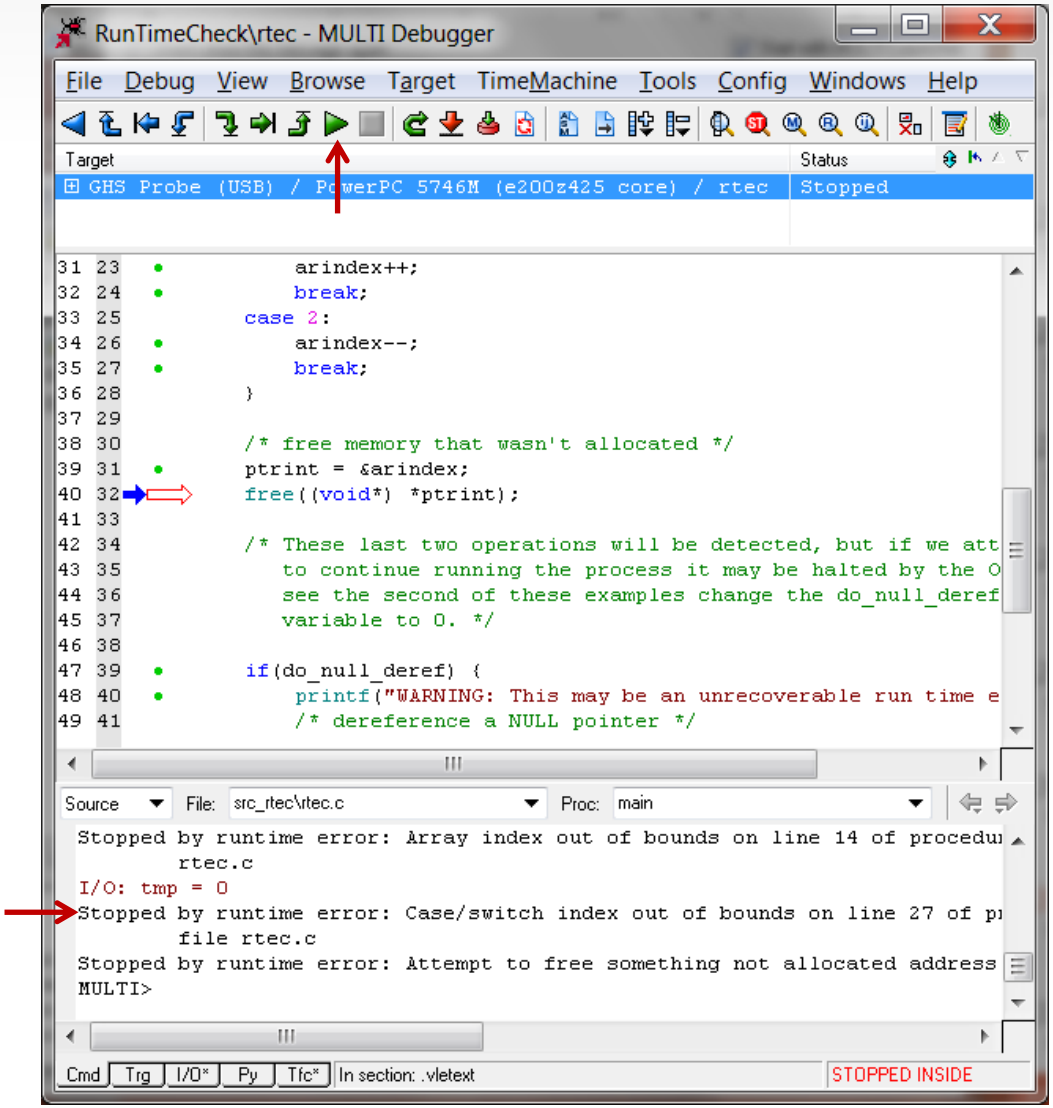
Hands-on 5.2.5: **Run Program and Note Second Error**

- Note that the program stops at Main:27 and reports an error in the command window
- **Left click** on the "**Go**" button



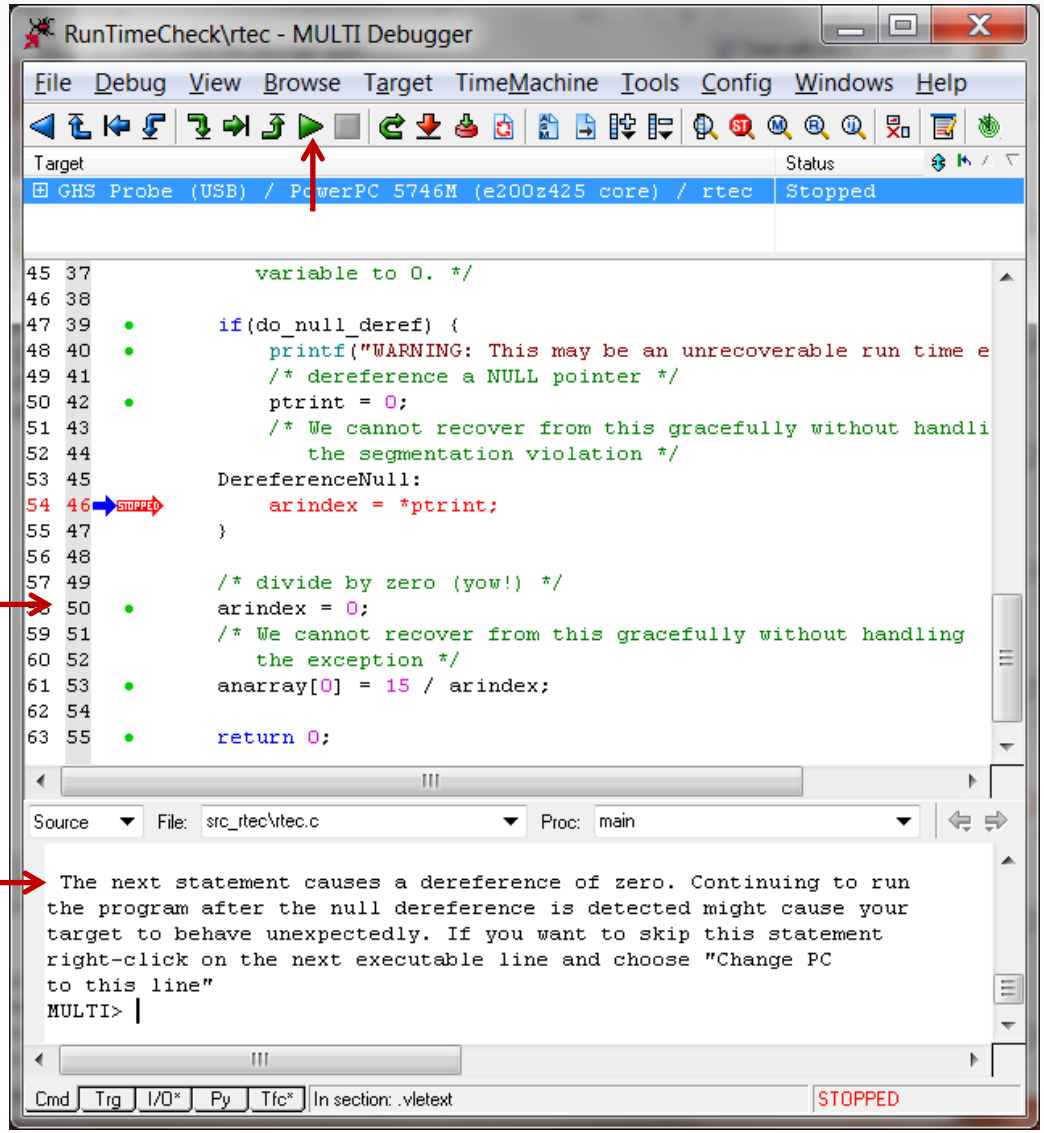
Hands-on 5.2.6: **Run Program and Note Third Error**

- Note that the program stops at Main:32 and reports another error
- **Left click** on the "**Go**" button



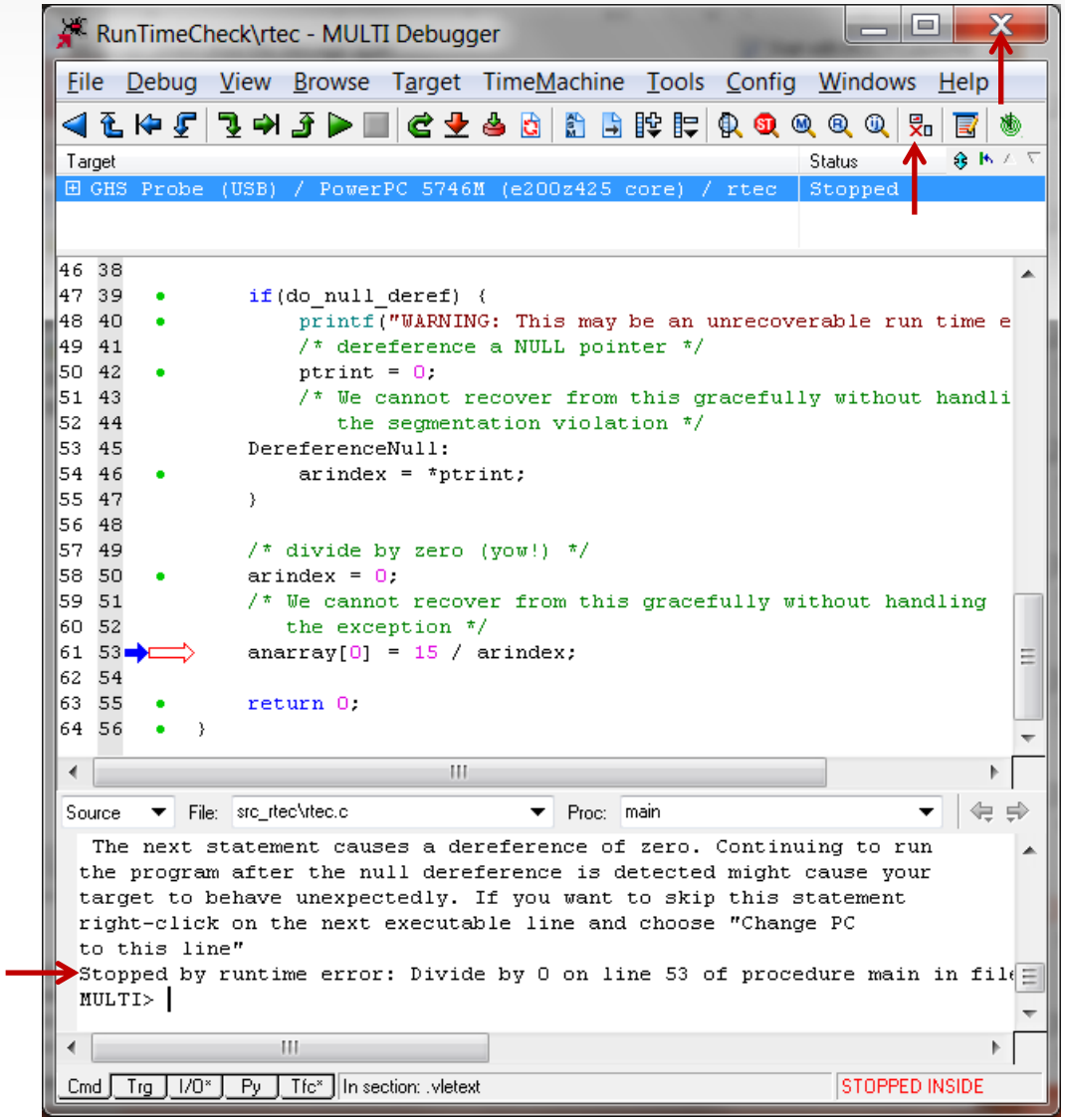
Hands-on 5.2.7: **Run Program and Note Fourth Error**

- Note that the program stops at Main:46 and reports an error
- Dereferencing the null pointer would cause an exception on this board
- **Right click** on main:50 and left click on “Change PC To This Line”
- **Left click** on the “**Go**” button



Hands-on 5.2.8: **Run Program and Note Fifth Error**

- Note that the program stops at Main:53 and reports an error
- **Left click** on the "**Disconnect**" button to close the debugging connection
- Close the Debugger window

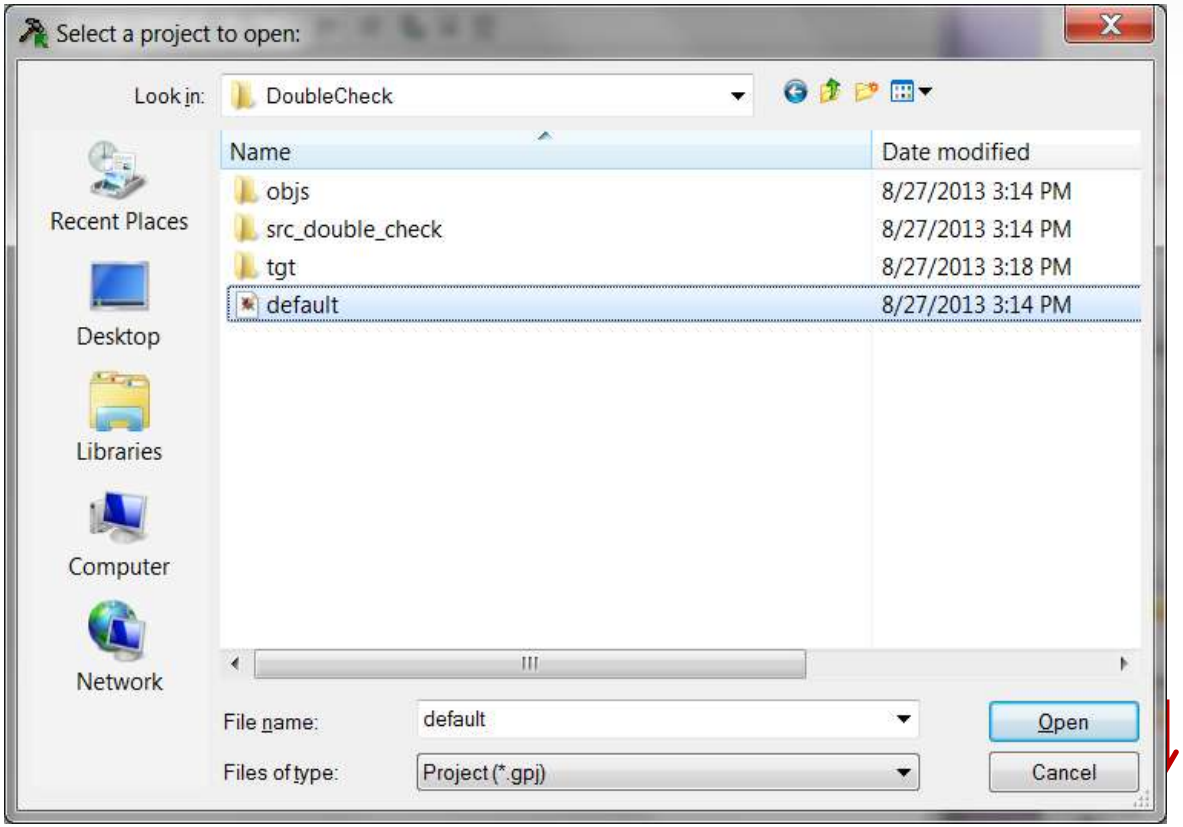


Hands-on 5:

- Advanced Tools
  1. TimeMachine Debugger
  2. Runtime error check
  3. DoubleCheck
  4. MISRA checking

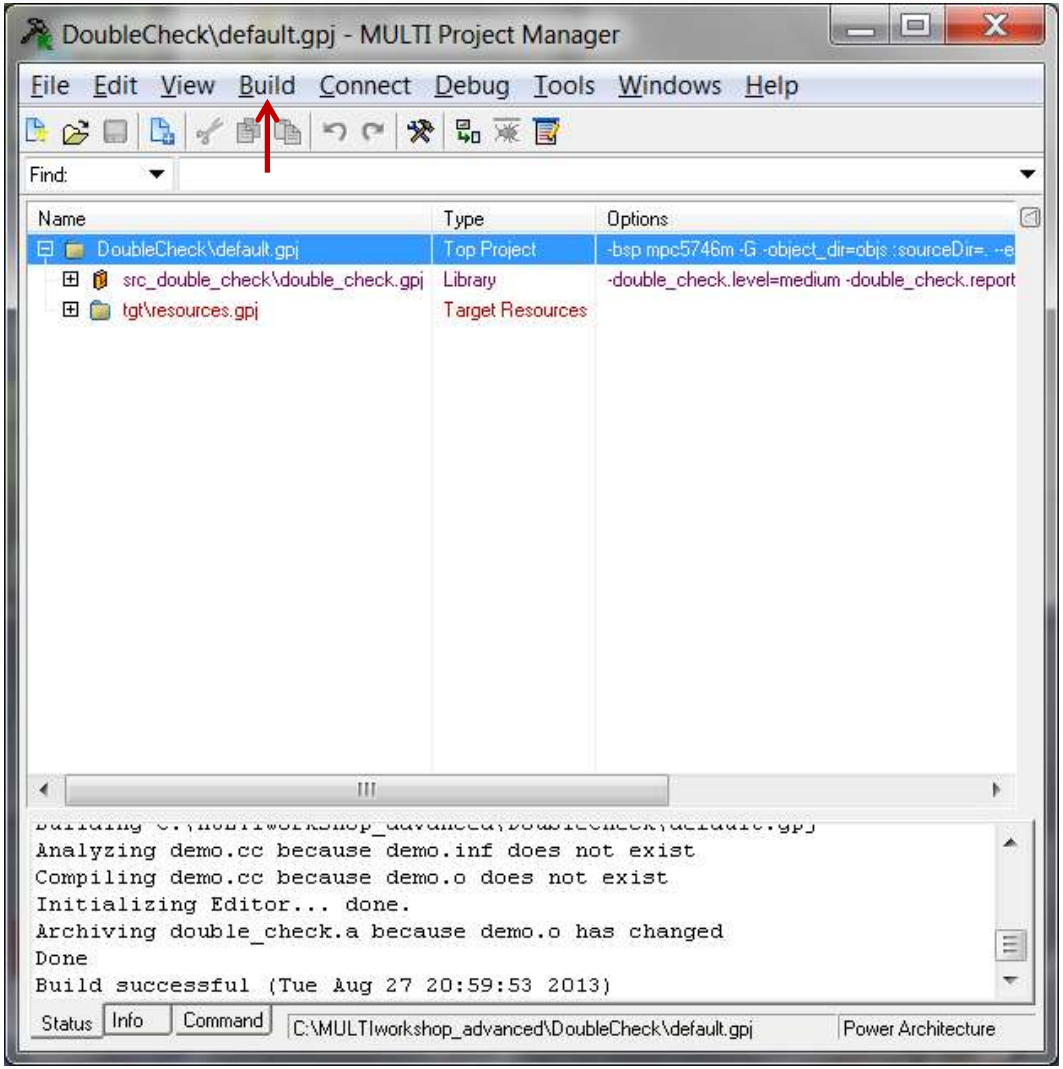
Hands-on 5.3.1: **Load the DoubleCheck Project**

- In **File** menu of Project Manager, left click on **Open Project**
- Navigate to **MULTIworkshop_advanced /DoubleCheck /default.gpj**
- **Left click** on the "**Open**" button



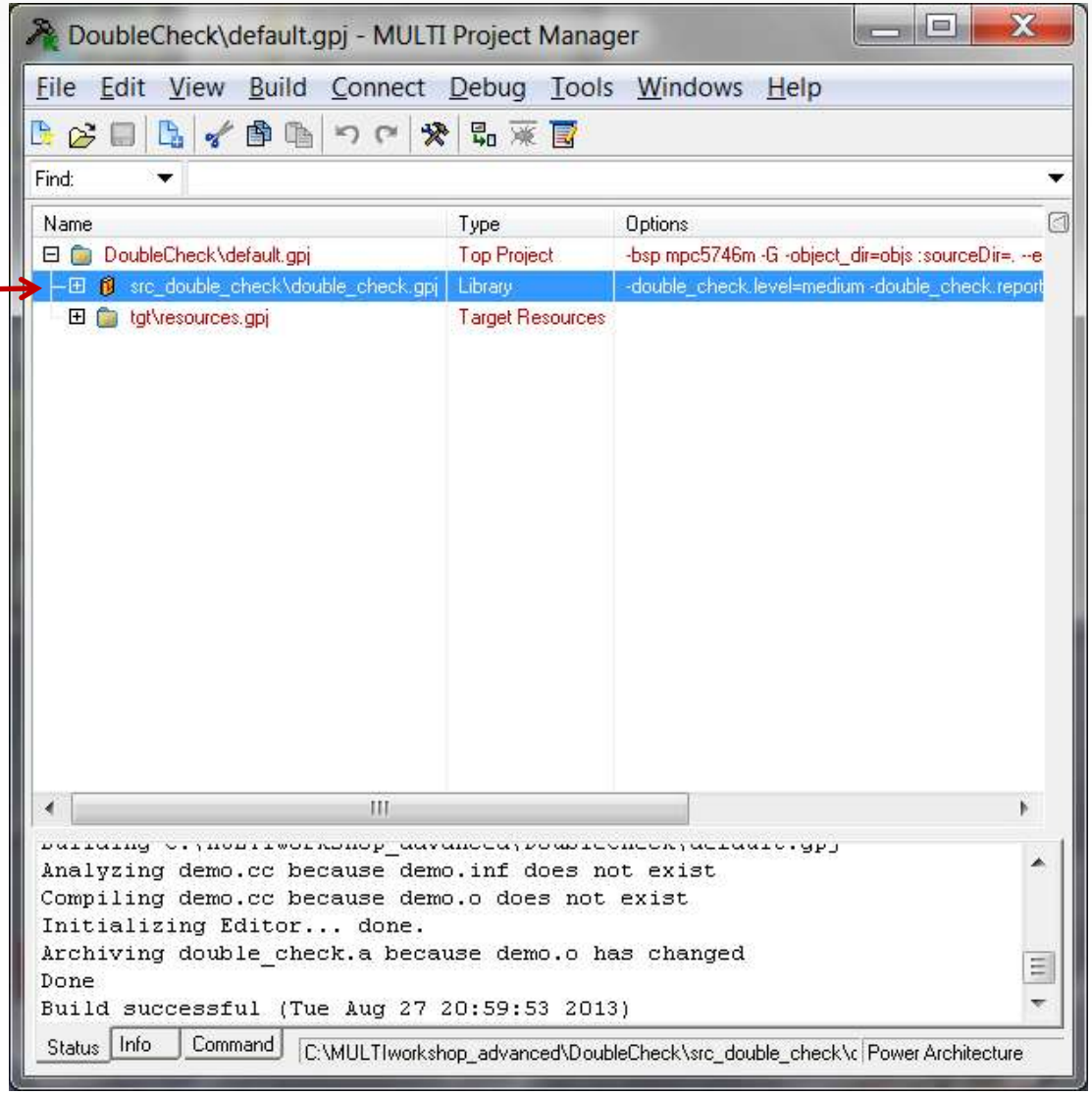
Hands-on 5.3.2: **Build the DoubleCheck Project**

- **Left click** the "**Build**" button in the Project Manager window
- **Close** the "Build Details" and "MULTI Editor" windows that pop up



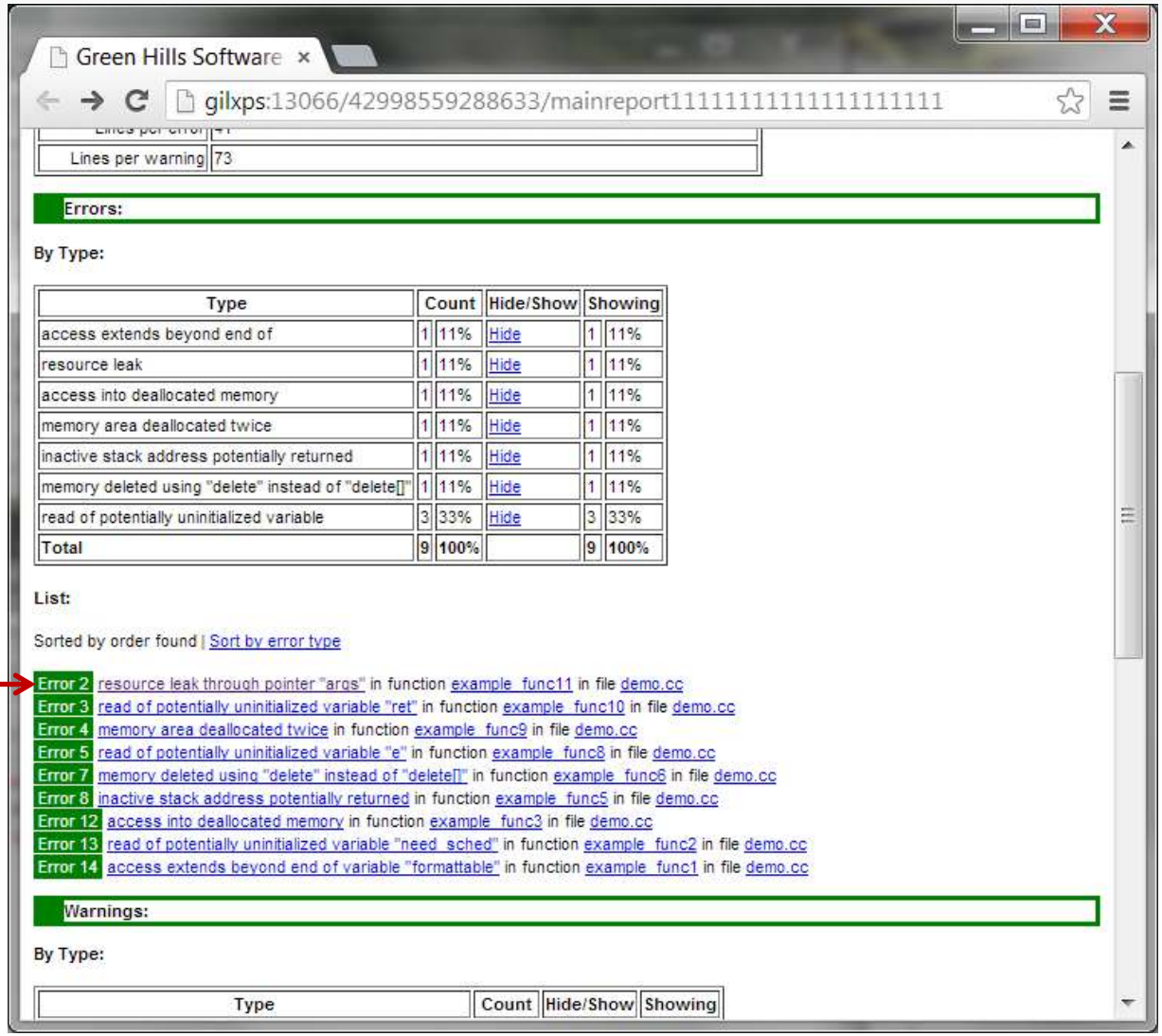
Hands-on 5.3.3: **Open the DoubleCheck Report**

- **Right click** on **double_check.gpj** in the Builder window
- **Left click** on "**View DoubleCheck Report**" in the popup menu



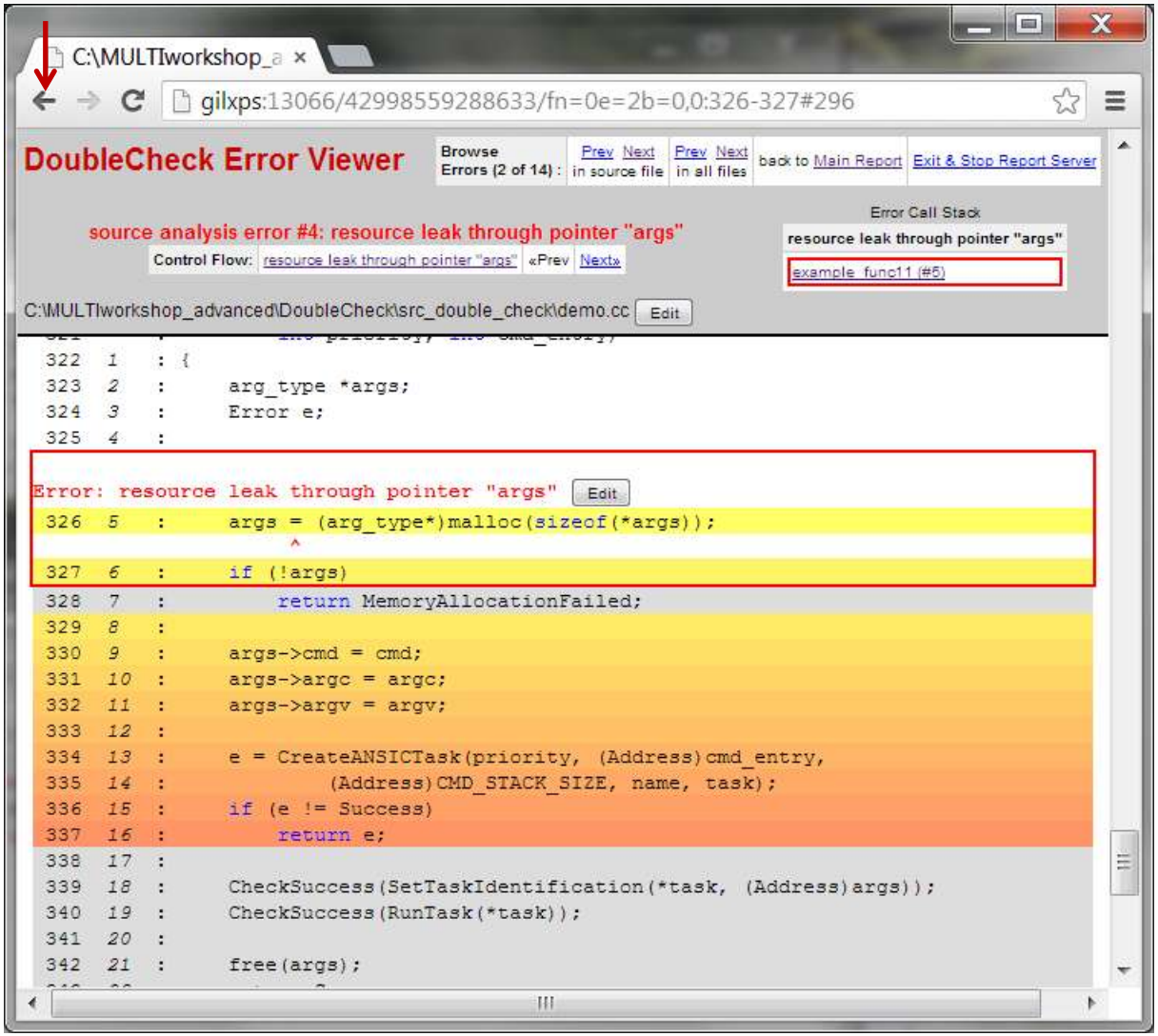
Hands-on 5.3.4: **Examine the DoubleCheck Report**

- **Scroll** down to the Errors section of the Browser window
- **Left click** on **“Error 2”**



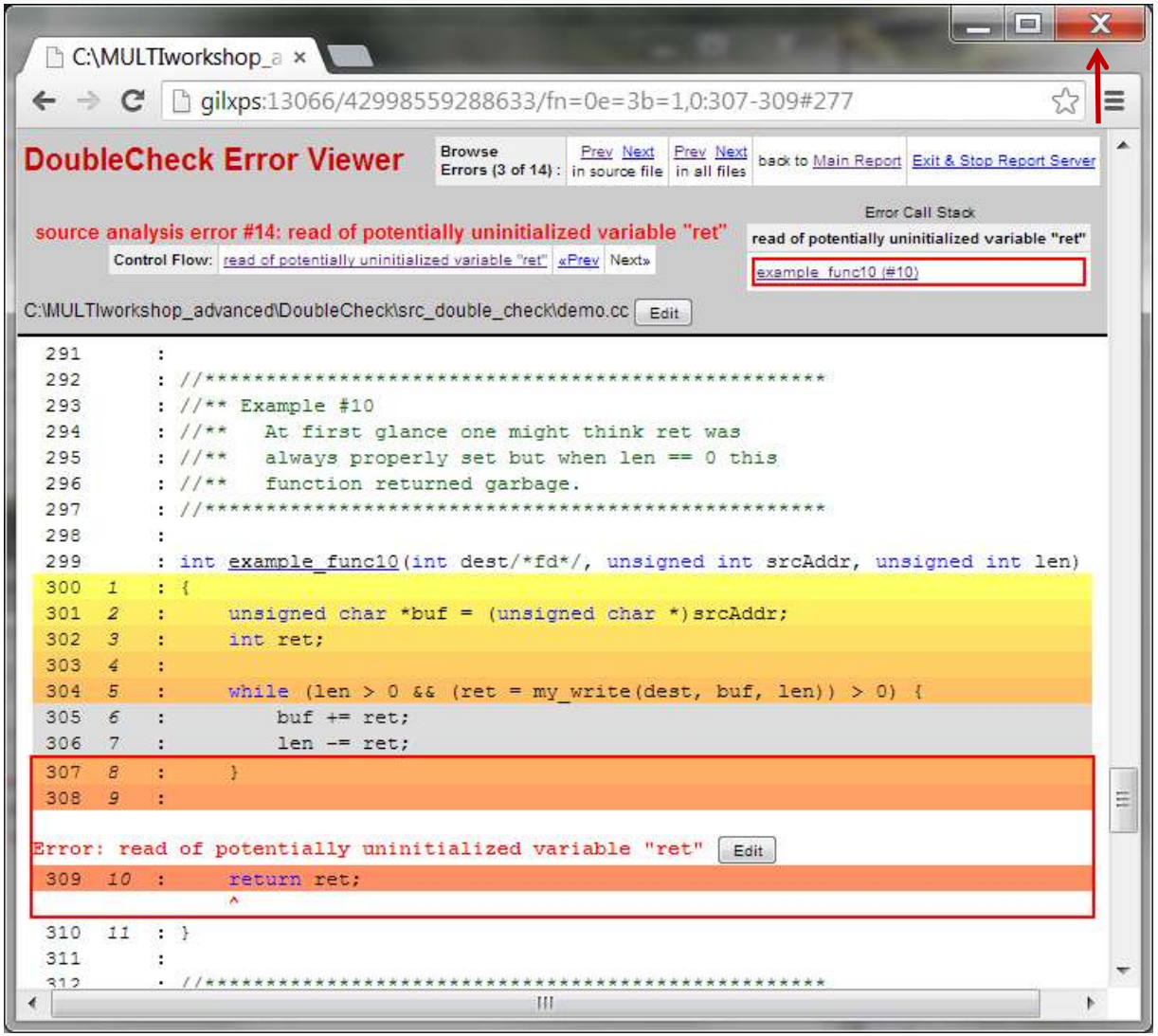
Hands-on 5.3.5: **Examine First DoubleCheck Error**

- **Scroll** down to the highlighted information in the Browser window.
- Note the resource leak of args in the code
- Left click on the "**Back**" button
- Left click on "**Error 3**"



Hands-on 5.3.6: **Examine Second DoubleCheck Error**

- **Scroll** down to the highlighted information in the Browser window.
- Note the uninitialized variable ret in the code
- **Close** the Browser window

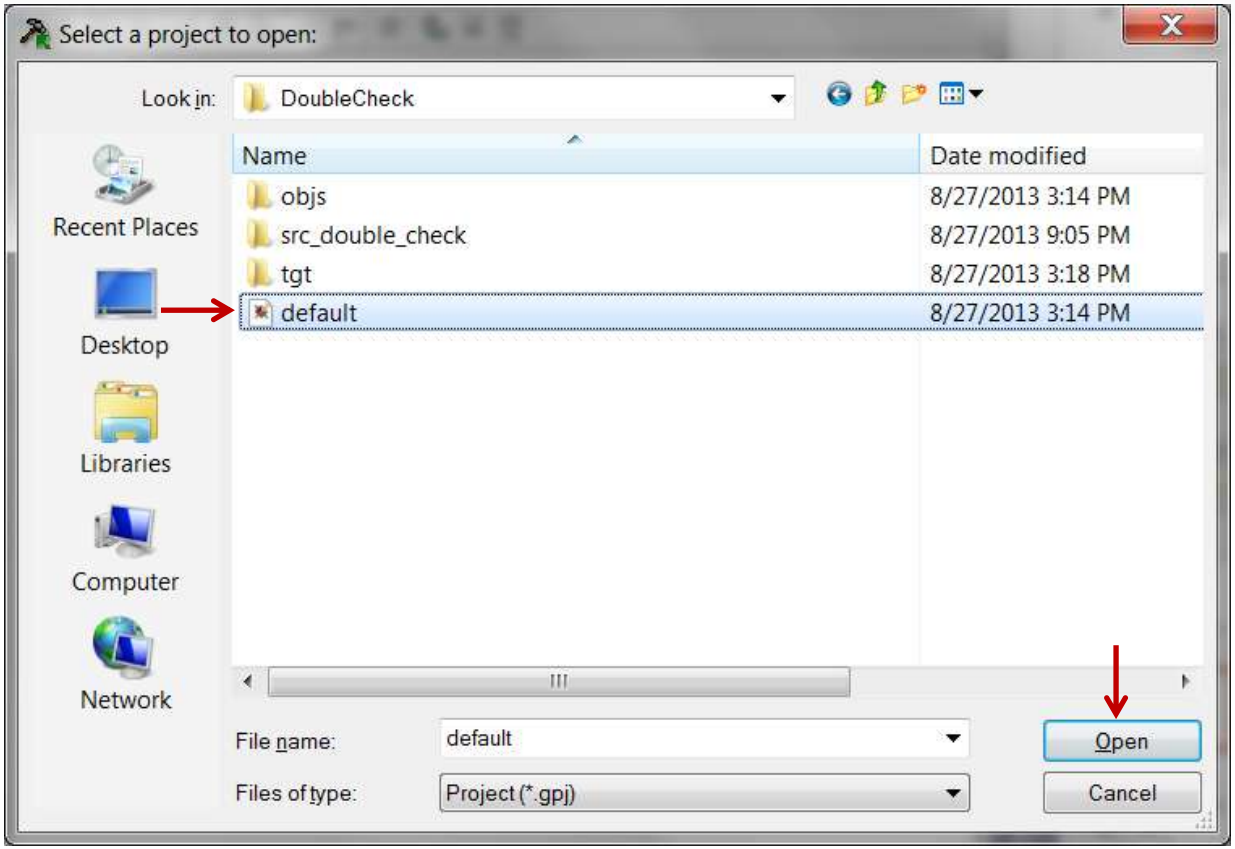


Hands-on 5:

- Advanced Tools
  1. TimeMachine Debugger
  2. Runtime error check
  3. DoubleCheck
  4. MISRA checking

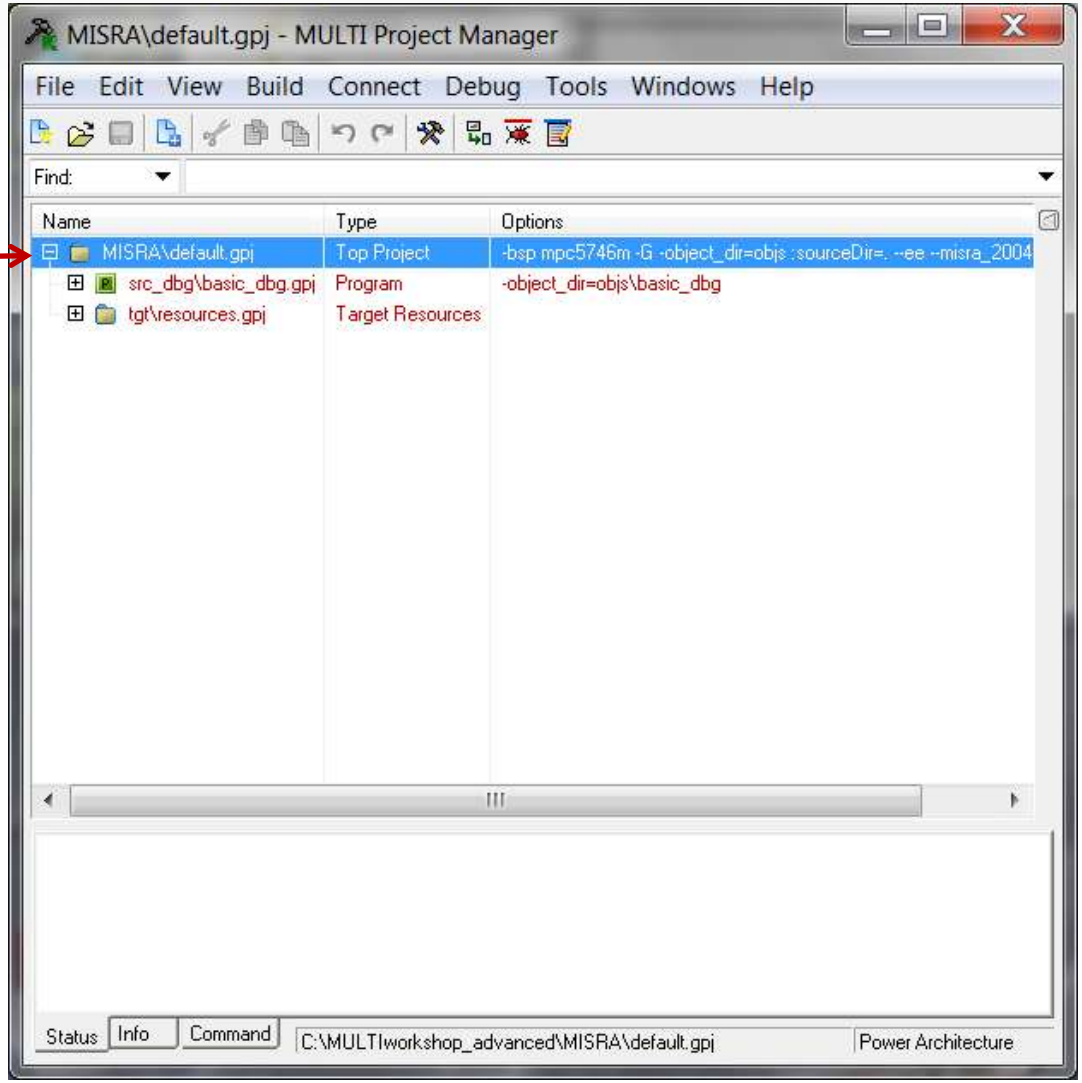
Hands-on 5.4.1: **Load the MISRA Project**

- In **File** menu of Project Manager, **left click** on **Open Project**
- Navigate to **MULTIworkshop_advanced\MISRA\default.gpj**
- **Left click** on the "**Open**" button



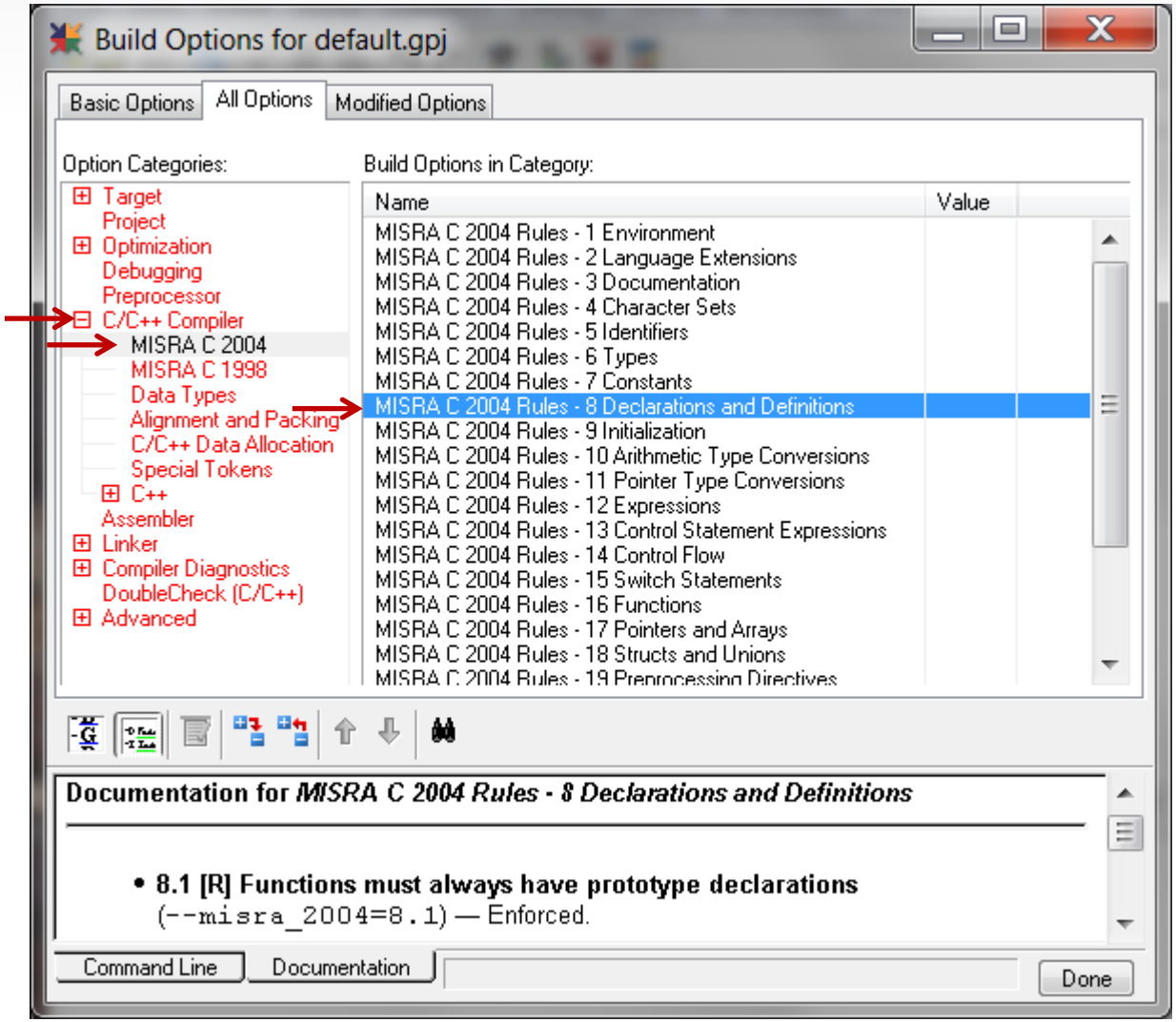
Hands-on 5.4.2: **View MISRA Rule Checking Options - 1**

- **Right click** on **default.gpj** in the Project Manager window
- **Left click** on "**Set Build Options…**" in the popup menu



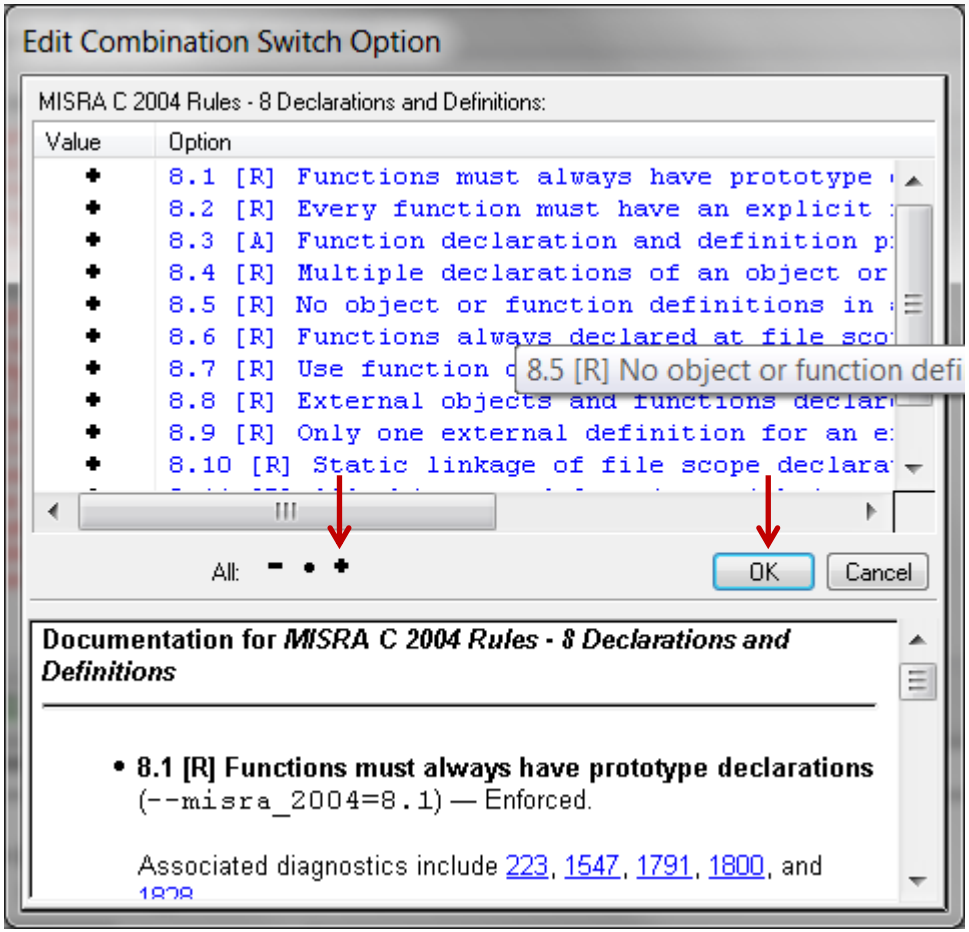
Hands-on 5.4.3: **View MISRA Rule Checking Options - 2**

- Expand the C/C++ Compiler options
- **Left click** on "**MISRA C 2004**"
- **Right click** on **group 8** (Declarations and Definitions)
- **Left click** on **"Set MISRA C** 2004 Rules" in the popup menu



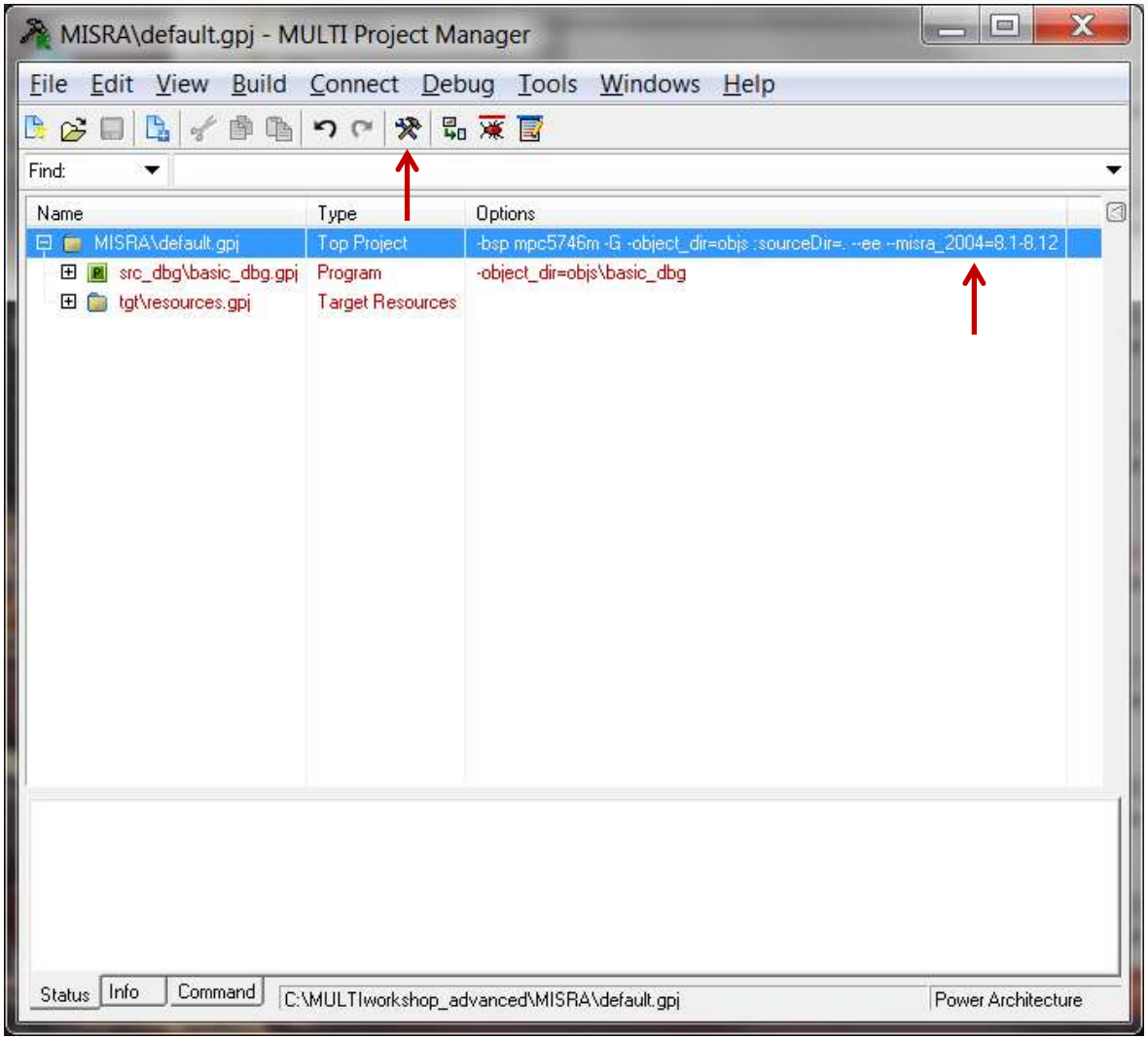
Hands-on 5.4.4: **Set MISRA Rule Checking Options**

- **Left click** on the "+" button
- Note that all of the group 8 checks are enabled
- **Left click** on "OK"
- **Left click** on "Done" in the Build Options window



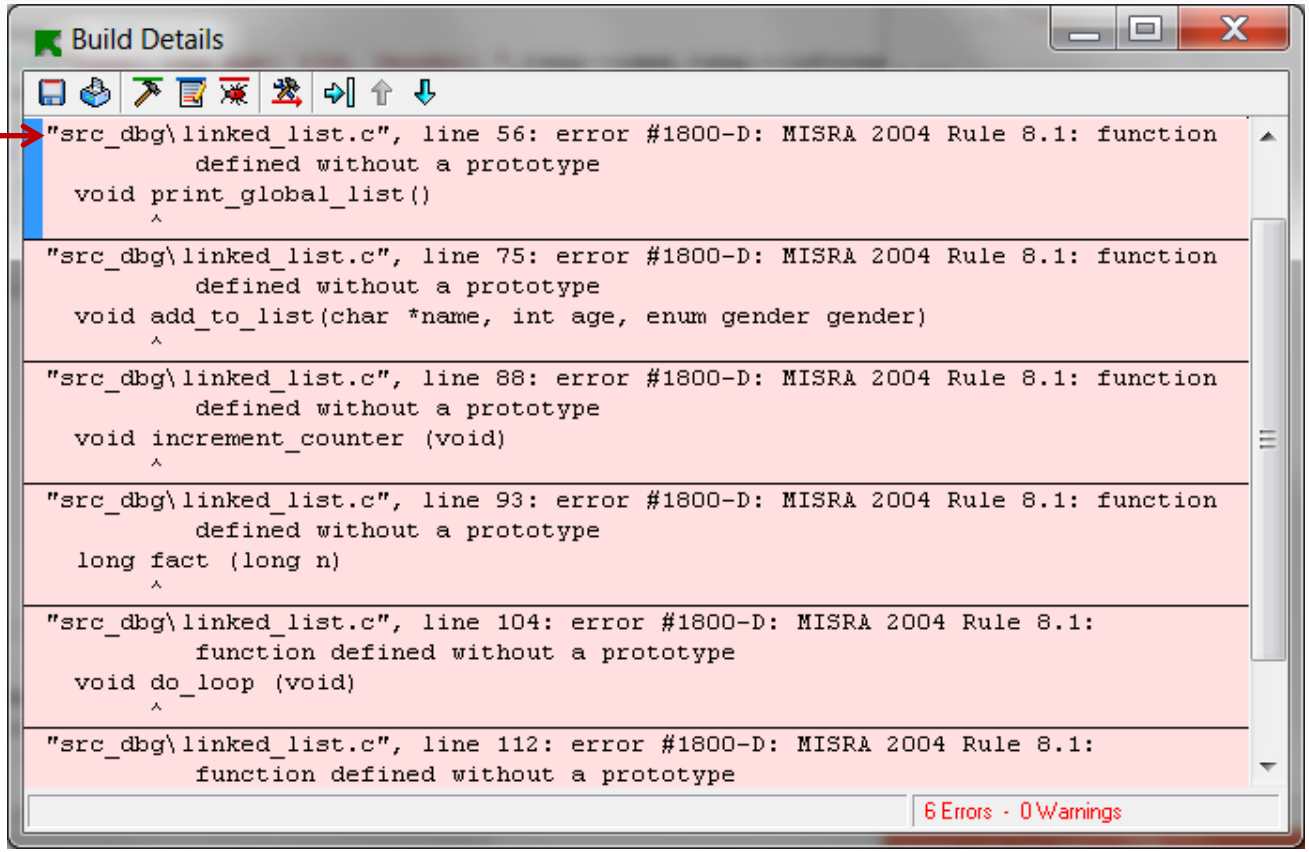
Hands-on 5.4.5: **Build with MISRA Rule Checking**

- Note that the MISRA checks have been added to default.gpj in the Project Manager window
- **Left click** on the “Build” button



Hands-on 5.4.6: **View MISRA Rule Violations**

- **Left click** on the first error in the Build Details window to see the corresponding source code in the Editor window
- **Close** the Build Details and Editor windows.



Additional Resources

- Freescale MCU Training and Events
  - Summary: Live in-depth training, seminars, on-demand training, webcasts and other resources.
  - Website: www.freescale.com/training

- Green Hills MULTI Training Class
  - Summary: This two-day course provides complete exposure to the MULTI® Integrated Development Environment (IDE), teaching students how to fully tap in to all of its powerful capabilities while developing their applications.
  - Website: http://www.ghs.com/training.html
  - Contact: Mark Wagner (mwagner@ghs.com)

Q&A

- Thank you


freescale™