# **Armstrap Documentation**

Release 0.0.1

**Charles Armstrap** 

Mar 20, 2017

## Contents

1	Introduction								
2	Hardware Overview         2.1       Armstrap Eagle	<b>5</b> 5							
3	Getting Started with C/C++ Development Tools for Armstrap Boards, Eclipse Edition3.1Overview3.2Download and Installation3.3Consolidate all Downloaded Parts into a Single Folder3.4Configuring C/C++ Development Tools for Armstrap boards, Eclipse Edition, for First Use3.5Creating a C/C++ Project3.6Creating a Build of a C/C++ Project3.7Downloading and Debugging Code	9 9 9 10 12 17 28							
4	Armstrap Naming and Versioning Convension4.1Format4.2Rules4.3Remarks4.4Workflow Example4.5Naming Examples	<b>39</b> 39 40 40 40 40							
5	Indices and tables	43							

Contents:

# CHAPTER 1

## Introduction

Armstrap a community of engineers and makers determined to help make ARM prototyping easy and fun. We focus on ease-of-use, helping real-world engineers bootstrap interactive objects or environments.

Armstrap boards are open-source electronics prototyping platforms centered on the powerful STM32F4 ARM processor. Each board can be built by hand or purchased preassembled; the software can be downloaded for free. The hardware reference designs (CAD files) are available under an MIT license, you are free to adapt them to your needs.

Our Goals:

- To provide an easy-to-use ARM development experience on Microsoft Windows, Apple Macs and Ubuntu Linux platforms.
- To provide open-source ARM development that enables real-world engineers to bootstrap any project.
- To have fun!

# CHAPTER 2

## Hardware Overview

## **Armstrap Eagle**





Shipment Includes:

• One Armstrap Eagle Board

Standard Features

- 168MHz STM32F4 Cortex-M4 Microcontroller
- 512KB 2048KB on-chip flash
- 192KB 256KB RAM
- 2MB 8MB SPI flash for easy data and configuration storage
- On-board JTAG controller for easy flashing and debugging (with no code-size limitations)
- 35 digital input/output pins
- Ready-to-use realtime clock with backup battery
- Debug over a standard USB cable
- Serial Console for runtime communication and diagnostics
- Reset and Device Firmware Upgrade button

- User-programmable LED and button
- Powered via the debug USB connector or by the 5V +Vin line
- Breadboard friendly for quick prototyping
- Flexible development options (develop with one version and deploy with another)
- Schematics and board layout file are open source (via MIT License) allowing you to build and remix the board, either personally or commercially.

Armstrap Eagle 512 Features

- 168MHz STM32F407VET6 Cortex-M4 Microcontroller
- 512KB on-chip flash
- 192KB RAM
- 2MB SPI flash

Armstrap Eagle 1024 Features

- 168MHz STM32F417VGT6 Cortex-M4 Microcontroller
- 1024KB on-chip flash
- 192KB RAM
- 4MB SPI flash

Armstrap Eagle 2048 Features

- 168MHz STM32F427VIT6 Cortex-M4 Microcontroller
- 2048KB on-chip flash
- 256KB RAM
- 8MB SPI flash

Requires (sold separately):

- Mini-USB Cable for flashing and debugging the onboard ARM chip
- CR1216 battery for realtime clock backup
- FTDI TTL Serial Cable for serial console access (TTL-232R-3V3)
- GPIO through-hole header soldering
- A Microsoft Windows, Apple Mac or Ubuntu Linux computer for project development

# CHAPTER 3

## Getting Started with C/C++ Development Tools for Armstrap Boards, Eclipse Edition

### **Overview**

This guide outlines how to create C/C++ projects using the C/C++ Development Tools for Armstrap boards using Eclipse, build an ELF executable from your project source code, run and debug the executable on your Armstrap target.

While this specific document shows screenshots for Apple Mac OSX, the steps are verified to work on both Microsoft Windows and Ubuntu Linux machines as well.

## **Download and Installation**

- 1. Install Java (Java SE 6 or greater is recommended), which you can download at http://www.java.com/getjava.
- 2. Download "Eclipse IDE for C/C++ Developers" from http://www.eclipse.org/downloads
- 3. Download "GNU Tools for ARM Embedded Processors" from https://launchpad.net/gcc-arm-embedded
- 4. Download "Armstrap blinky examples" from https://s3.amazonaws.com/armstrap-public/examples/armstrap\_ blinkyexamples\_1.0.0.zip

## Consolidate all Downloaded Parts into a Single Folder

- 1. Create a folder on your Desktop called armstrap
- 2. Extract your downloaded "Eclipse IDE for C/C++ Developers" into <user>/Desktop/armstrap/eclipse
- 3. Extract your downloaded "GNU Tools for ARM Embedded Processors" into <user>/Desktop/armstrap/gccarm
- 4. Extract your downloaded "Armstrap blinky examples" into <user>/Desktop/armstrap/workspace

# Configuring C/C++ Development Tools for Armstrap boards, Eclipse Edition, for First Use

Complete the following steps to configure C/C++ Development Tools for Armstrap boards, Eclipse Edition, for first use:

- 1. Launch Eclipse by clicking on the <user>/Desktop/armstrap/eclipse/eclipse executable
- 2. When prompted, select the *<user>/Desktop/armstrap/workspace* folder in which to store Eclipse projects and click **OK**.

00	Workspace Launcher							
Select a workspace								
Eclipse stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.								
Workspace: /Users/armstrap/Desk	top/armstrap/workspace r Browse							
	•							
Use this as the default and do not ask again								
	Cancel OK							

Tip: Enable Use this as the default and do not ask again to save a project folder as your default workspace.

3. In the Eclipse welcome screen, select the Workbench icon on the far right to open the workbench view.



4. Eclipse highlights the active perspective on the perspectives bar, as shown in the following image. The first time you use Eclipse the workbench view opens in the C/C++ perspective.

00	C/C++ - Eclipse - /Users/grmstrap/D	esktop/armstrap/works	pace	
i 📬 • 🗏 🦷 è   🎭 - < +i 👌 • 😂 • 🕞 • 🥝	) +   🏷 + 🔕 + 🗞 - 🕲 🧀 🔗 +   🗉	I III Ja • Pi • ← ← •	⇒ + 📄 🔍 Quick	Access
Project Explorer 🛛 🗖 🗖				Eo ⊠ "2 □
E 45 D V				An outline is not available.
0	Problems 🕱 🕢 Tasks 📃 Console 🔲 Pr	operties		
D	Description	Resource Path	Location	Туре

5. Debugging Armstrap requires the "C/C++ GDB Hardware Debugging" plugin. To install the plugin, select the Help>>Install New Software... menu item

Eclipse	File	Edit	Source	Refactor	Navigate	Search	Project	Run	Window	Help		
		20					armstra	an			Search	
000					C/C++ - E	clipse – /l	Jsers/arms	trap/D	esktop/arm		Welcome	Ľ
1 🕄 • 🔚 🕇	ñ 📥 🖇	• 5	· 💼 • 😂	• 😋 • 😋 •	\$ו <b>Q</b> • 8	• 🏊 • 🕅	N 🥭 🗁 /	🔗 • 🛛 🗖	<b>11   2 -</b> 1		Under Comstants	]C/C++
	Evolore										<sup>(2)</sup> Search	
- Flojeci	Explore	~		~							Dynamic Help	~
		ie	⊐ <del>*</del> >   •.								Key Assist OʻHL Tips and Tricks & Report Bug or Enhancement Cheat Sheets	ble.
									•	<	Eclipse Marketplace Check for Updates Install New Software	

- 6. Configure the plugin installation
  - In the **Work with:** drop-down, select the version of Eclipse you downloaded (for example "Kepler"). As seen in label mark 1 in picture.
  - In the search field, type Hardware as seen in label mark 2 in picture.
  - Click the check-box to select the C/C++ GDB Hardware Debugging plugin as seen in label mark 3 in picture.
  - Click the **Next** button, as seen in label mark 4 in picture and accept the licensing agreement to complete the installation.
  - You will need to restart Eclipse when the plugin installation is complete.

● ○ ● Ir	Istall
Available Software Check the items that you wish to install.	
Work with: Kepler - http://download.eclipse.org/releases/kepler	• 1 • Add
2	Find more software by working with the " <u>Available Software Sites</u> " preferences.
Hardware	(3)
Name 3	Version
C/C++ CDB Hardware Debugging	7 2 0 201300180223
	TEIDEOLOGIOTEO
L	
Select All Deselect All 1 item selected	
Details	
Support for launching GDB to do hardware debugging.	
	More
Show only the latest versions of available software	Hide items that are already installed
Group items by category	What is <u>already installed</u> ?
Show only software applicable to target environment	
Contact all update sites during install to find required software	
	4
(?)	< Back Exect > Cancel Finish

## Creating a C/C++ Project

Complete the following steps to create a C or C++ project in C/C++ Development Tools for Armstrap boards

- 1. Switch to the C/C++ perspective.
- 2. Select File>>New>>C Project to open the New Project Wizard.

Eclipse	File Edit Source Ref	actor Navigate	Search Project Run Window Help
	New Open File Close Close All	<b>て第N</b> 第₩ 企業W	Makefile Project with Existing Code C++ Project C Project Project Project
ib Project	I Save I Save As I Save All Revert	쁐S 쇼쁐S	<ul> <li>Convert to a C/C++ Autotools Project</li> <li>Convert to a C/C++ Project (Adds C/C++ Nature)</li> <li>Source Folder</li> <li>Folder</li> </ul>
	Move P Rename Refresh Convert Line Delimiters	F2 F5 To	C Source File Header File File from Template Class Task
	Print Switch Workspace Restart	₩P	nusk المعادي ال المعادي المعادي
	≥ Import ≧ Export		
	Properties	<b>%</b> I	

- 3. Configure the C Project
  - Give the project the name *blinky*, as seen in label mark 1 in picture.
  - Under **Project Type**, select the **Empty Project** option, as seen in label mark 2 in picture.
  - Under Toolchains, select Cross GCC option, as seen in label mark 3 in picture.

● ○ ○	C Project
C Project Create C project of selected type	
Project name: blinky Use default location Location: /Users/armstrap/Desktop/arm	nstrap/workspace/blinky Browse
Choose file system: default	÷ <b>3</b>
Project type: Constraint of the second seco	Toolchains: Cross GCC MacOSX GCC
Show project types and toolchains on	y if they are supported on the platform
(?) < Back	Next > Cancel Finish

- 4. Click Next to open the Select Configuration page.
- 5. Enable **Debug** to configure the project to allow debugging your executable, and/or enable **Release** to configure the project to allow building a smaller, faster executable optimized for release. Note: For purposes of this tutorial, ensure you enable **Debug**.



- 6. Click Next to open the Cross GCC Command page.
- 7. In the **Cross compiler prefix** text box, enter *arm-none-eabi-*, including the hyphen (-) at the end, to specify the correct compiler for Armstrap targets.
- 8. In the Cross compiler path text box, browse to the location of the *<user>/Desktop/armstrap/gcc-arm/bin* directory to specify the location of the compiler.



- 9. Click Finish to create your project and return to the workbench view.
- 10. Verify your project source code appears in the Project Explorer. If it does not, you may have to hit the **F5** key to refresh.

00	C/C+	+ - blinky/source/main.c - Eclipse - /Users/arm	strap/Desktop	/armstrap/work	space	R5
📑 - 🖪 🖻 📥 🛞 - 🏀 - 💼 🗎	8	🔲 🗊 📸 • 🗳 • 🗳 • 🗇 • 🏇 • 🕥 • 🎭 • 🙆	🖢 💪 🔗 📲 🖄	$\bullet \boxtimes \bullet \leftarrow \bullet$	×	
					Q Quick	Access
Project Explorer 🛿		i main.c 🕄			- 0	
<ul> <li>blocky</li> <li>includes</li> <li>cripts</li> <li>cri</li></ul>	p •	<pre>#include main.n const uint16_t LEDS = GPI0_Pin_1; const uint16_t LED[1] = {GPI0_Pin_1}; const long LED_COUNT = 1; e int main() {     init();     do     {         loop();     }     while (1);     return 0; // never reached } e void loop() {     uint32_t counter = 0;     while (1) {         counter+;         GPI0_ResetBits(GPI0C, LEDS);         GPI0_ResetBits(GPI0C, LEDS);         delay(250);     }; </pre>	• % 2]);			<pre>% *** * * * * * * * * * * * * * * * * *</pre>
		0 items				
		Description	Resource	Path	Location	Туре
🖻 blinky				1		

In the next section of this tutorial, you create an executable build of your project to enable it to run.

## Creating a Build of a C/C++ Project

Before you can run your project, you need to test that your source code compiles by creating an executable build of your project. Complete the following steps to create an executable build of a C/C++ project:

- 1. Switch to the C/C++ perspective.
- 2. Right-click (or Ctrl-click on a Mac) your project in the Project Explorer tab and select Properties.



3. Select C/C++ Build>>Settings in the left pane of the Properties dialog box. Verify that Cross Settings>>Tool Settings>>Prefix is set to *arm-none-eabi-* and Cross Settings>>Tool Settings>>Path is set to the bin path to your compiler toolchain <*user*>/Desktop/armstrap/gcc-arm/bin



#### 4. Under Cross GCC Compiler>>Symbols>>Defined symbols, enter





#### 5. Under Cross GCC Compiler>>Includes>>Include paths, enter

```
"${workspace_loc:/${ProjName}/includes/CMSIS}"
"${workspace_loc:/${ProjName}/includes/STM32F4xx}"
"${workspace_loc:/${ProjName}/includes/STM32F4xx_StdPeriph_Driver/inc}"
```



#### 6. Under Cross GCC Compiler>>Miscellaneous>>Other flags, enter

-c -fno-common -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -MD



#### 7. Under Cross GCC Linker>>Libraries>>Library search path, enter

<pre>["\${workspace_loc:/\${ProjName}/scripts}"</pre>	



#### 8. Under Cross GCC Linker>>Miscellaneous>>Linker flags, enter

-Tstm32\_flash.ld -nostartfiles -Wl,--gc-sections -mthumb -mcpu=cortex-m4 -mthumb -→mfloat-abi=hard -mfpu=fpv4-sp-d16



#### 9. Under Cross GCC Assembler>General>>Assembler flags, enter

-mcpu=cortex-m4 -mthumb



10. In the Build Artifacts tab, under Artifact extension, enter

#### elf

000	Properties for b	linky
type filter text	Settings	Ģ••⊋•• ▼
▶ Resource Builders ♥C/C++ Build Build Variables Environment	Configuration: Debug [Active]	Manage Configurations
Logging Settings Tool Chain Editor	🛞 Tool Settings 🛛 🎤 Build Steps	Build Artifact Binary Parsers 📀 Error Parsers
►C/C++ General Project References Run/Debug Settings	Artifact Type: Executable	•
Task Repository WikiText	Artifact extension: elf	▼ ▼
	Output p. fin	•
		Restore Defaults Apply
?		Cancel OK

11. Select C/C++ Build>>Tool Chain Editor in the left pane of the Properties dialog box. Set Current builder to *CDT Internal Builder* 



- 12. Click Apply and then OK to close the Properties dialog box.
- 13. Click the build icon in the toolbar or select **Project>>Build Project** in the workbench view to create an elf executable of your project. Verify your project builds successfully.

000 C/C+	-+ - blinky/source/main.c - Eclipse -	- /Users/armstrap/Desk	top/armstrap/w	orkspace		LL <sup>2</sup>
	ं 🗉 🖬 💼 र 😂 र 💽 र 🚱 र 🏇 र 🕗	• 💁 💁 🖕 🔗 •	⊿・招・← ←	×⇒× ≝		
				Q Quick A	ccess	C/C++
Project Explorer S       Image: Solution of the system       Image: Solution of the system<	<pre>     main.c 23     #include "main.h"     const uint16_t LEDS = GPI0.     const uint16_t LED[1] = {Gi     const long LED_COUNT = 1;         init();         uint();         Building project         Building project         Invoking Command: arm-none-eabi-gcc         Always run in background         Cancel         Counter++;         GPI0_ResetBits(GPI0         delay(250);         }; </pre>	<pre>Pin_1; PIO_Pin_1}; ild Project -DSTx_StdPeriph_Driver/st Details &gt;&gt; F C, LEDS); LED[counter % 2]);</pre>	rc/stm32f4xx_tim. Run in Background		E O S "2 Main.h C LEDS : const C LED_COUNT Main0: int C LED_COUNT Main0: int I loop0: void delay(uint32	• ¥t ulint16_t int16_t(] : const long _t) : void
	📳 Problems 🕄 🖉 Tasks 📃 Conse 0 items	ole 🔲 Properties			69	▼ -  -  -
	Description	A Resource	Path	Location	Туре	
				Build Project: (12	%)	

14. The Console tab displays Build Finished if the build completes successfully, as shown in the following image. In the next section of this tutorial, you prepare to run and debug the ELF executable on your Armstrap target.

## **Downloading and Debugging Code**

Before you can run the ELF executable you created in the previous section on your Armstrap target, you need to create a Debug Configuration.

1. In the C/C++ perspective, select the Debug Configurations... in the debug drop-down

<pre>Project Explorer X Project Explorer X Project</pre>	● ● ● ● C/0	C++ - blinky/source/main.c - Eclipse - /Users/armstrap/Desktop/armstrap/workspace	R <sub>2</sub>
<pre>Project Explorer X</pre> <pre>Project Explorer Project Explorer Explorer Project Explorer Project Explorer Project Explorer Explorer Project Explorer Project Explorer Expl</pre>	🖆 • 🖪 👘 🍐   ≫ • ≪ • 🖬   🔌   ∠	○ □ □ ②・③・ ●・ ●・ ●・ ●・ ●・ ●・ ●・ ●・ ●・ ●・ ●・ ●・ ●・	Access
Image: Problems       Image: Tasks       Image: Console 83       Properties         CDT Build Console (blinky)       Image: Console 74, Cons	Project Explorer №         Image: Support State         Image:	<pre>Main.c E3</pre>	BE 0 S3       "2       □         Imain.h       □ LEDS : const uint16_t1       □         □ CLED : const uint16_t1       □ COUNT : const long         □ CLED : const uint16_t1       □ cop0 : void         □ init0 : void       □ delay(uint32_t) : void
CDT Build Console [blinky] umm-none-eabi-gcc - DSTM32F4 - DARM_MATH_CM4 - DUSE_STDPERIPH_DRIVER - I/Users/armstrap/Desktop/armstrap/works arm-none-eabi-gcc - DSTM32F4 - DARM_MATH_CM4 - DUSE_STDPERIPH_DRIVER - I/Users/armstrap/Desktop/armstrap/works ar		🗜 Problems 🖉 Tasks 🖳 Console 🕱 🔲 Properties 🕹 😯 😒 💷 🔠	= 🔓 🛃 🗐 • 📬 • 🗖 🗖
		CDT Build Console [blinky] Umm-none-eabi-gcc -DSTM32F4 -DARM_MATH_CM4 -DUSE_STDPERIPH_DRIVER -L/Users/armst arm-none-eabi-gcc -L/Users/armstrap/Desktop/armstrap/workspace/blinky/scripts -T 13:07:12 Build Finished (took 4s.969ms)	rap/Desktop/armstrap/works rap/Desktop/armstrap/works /startup_stm32f4xx.S rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works sap/Desktop/armstrap/works sap/Desktop/armstrap/works stm32_flash.ld -nostartfil

2. Double-click the **GDB Hardware Debugging** to create a new debug configuration. The debug configuration should be populated with settings from the current project.

000	Debug Configurations		
Create, manage, and run confi	gurations	Ŕ	r
Image: State of the state o	Configure launch settings from this dialog: <ul> <li>Press the 'New' button to create a configuration of the selected type.</li> <li></li></ul>	ıge.	
Filter matched 6 of 6 items			
(?)		Close Debug	

3. Change the debug configuration name to *blinky (Flash and Debug)*, as seen in label mark 1 in picture. Click **Enable auto build** in the **Build configuration** section to enable builds to automatically happen (if needed) when the debug button is pressed, as seen in label mark 2. Click the **Select other...** link, as seen in label mark 3, to configure the GDB Hardware Debugging Launcher.

● ○ ○ Debug Configurations				
Create, manage, and run confi	gurations			
Ype filter text         C/C++ Application         C/C++ Attach to Application         C/C++ Postmortem Debu         C/C++ Remote Applicaticic         C GDB Hardware Debugging         C blinky Debug         ▶ Launch Group	Name: blinky (Flash and Debug)			
Filter matched 7 of 7 items	Using Standard GDB Hardware Debugging Launcher - <u>Select</u> Apply Revert			
?	Close Debug			

4. Check **User configuration specific settings** option as seen in label mark 1 in picture. Select **Standard GDB Hardware Debugging Launcher** in the list of Launchers. Click the **OK** button to complete the GDB launcher configuration.

● ○ ○ _ Select Preferred Launcher	-
This dialog flows you to specify which launcher to use when multiple launchers are available for a configuration and launch mode.	- OC
Use configuration specific settings <u>Change Workspace Settings</u>	
Launchers: 💋 💋 🕹	
GDB (DSF) Hardware Debugging Launcher	
Standard GDB Hardware Debugging Launcher	
	reinst Brown
	browse
	Browse
Description	*
Jtag hardware debugging using the standard debugger Framework (CDI).	pplication'
3	
	ttings
Cancer	
Using GDB (DSF) Hardware Debugging Launcher - <u>Select other</u>	Apply Revert
	Close Debug

#### 5. In the Debugger tab,

- Under, **GDB Setup>>GDB Command**, enter the full path to the location of *arm-none-eabi-gdb* that was downloaded with GNU Tools for ARM Embedded Processors. This should be <*user>/Desktop/armstrap/gcc-arm/bin/arm-none-eabi-gdb*, as seen in label mark 1 in picture
- Under Remote Target, uncheck Use remote target, as seen in label mark 2.

● ○ ○ Debug Configurations			
Create, manage, and run confi	gurations		
Ype filter text         C/C++ Application         C/C++ Application         C/C++ Remote Applicatic         C/C++ Remote Applicatic         C/C++ Remote Applicatic         C/C++ Remote Applicatic         C/DB Hardware Debugging         C blinky Debug         ▶ Launch Group	Name: blinky (Flash and Debug) Main Debugger Startup Source Common GDB Setup GDB Command: [/Users/armstrap/Desktop/armstrap/gcc-arm/bin/arm-none-eat] Browse Variables Command Set: Standard (Mac OS) ‡ Protocol Version: mi ‡ Verbose console mode Remote Target Use remote target JTAG Device: Generic TCP/IP ‡ Host name or IP address: localhost Port number: 10000		
Filter matched 7 of 7 items	other Apply Revert		
?	Close Debug		

- 6. In the **Startup** tab, under the Initialization Commands:
  - Uncheck Reset and Delay (seconds) option
  - · Check Halt option
  - For Apple Mac OSX machines, enter the following start-up script

```
target extended /dev/tty.usbmodem7B4078B1
monitor swdp_scan
attach 1
monitor vector_catch disable hard
set mem inaccessible-by-default off
set print pretty
```

• For Ubuntu Linux machines, enter the following start-up script

```
target extended-remote /dev/ttyACM0
mon swdp_scan
attach 1
monitor vector_catch disable hard
set mem inaccessible-by-default off
set print pretty
```

· For Microsoft Windows machines, enter the following start-up script

```
target extended-remote \\.\COM2
mon swdp_scan
```

```
attach 1
monitor vector_catch disable hard
set mem inaccessible-by-default off
set print pretty
```

- Check Load image option and Use project binary
- Check Load symbols option and User project binary

● ○ ○ Debug Configurations				
Create, manage, and run configur	ations			
Ype filter text         C/C++ Application         C/C++ Atach to Application         C/C++ Postmortem Debugge         C/C++ Remote Application         C GDB Hardware Debugging         C blinky (Flash and Debug)         ▶ Launch Group	Name:       blinky (Flash and Debug)         Main       Debugger       Startup         Main       Debugger       Startup         Initialization Commands       Initialization Commands         Reset and Delay (seconds):       3         Halt       Halt         target extended /dev/tty.usbmodem7B4093D1         monitor swdp_scan         attach 1         monitor vector_catch disable hard         set men inscretelia_bw_default off         Load image         Use project binary:         blinky.elf         Use file:         Image offset (hex):         Load symbols         Use project binary:         blinky.elf         Use project binary:         blinky.elf         Use file:         Use file:         Use file:         Use file:	Workspace File System		
Filter matched 7 of 7 items	Using Standard GDB Hardware Debugging Launcher – <u>Select other</u>	Apply Revert		
(?)		Close Debug		

- 7. Click the **Apply** button and the **Close** button to return to the C/++ perspective.
- 8. Open *main.c* from in the project find the first line inside the main() function. In this project, the first line is a call to init(). Right-click (or Ctrl-click on a Mac) on the margin to open a menu-item and select the **Toggle Breakpoint** menu option to set a breakpoint.

● O O C/C+	+ - blinky/source/main.c - Eclipse - /Us	ers/armstrap/Desk	top/armstrap/v	vorkspace	R <sub>2</sub>
📬 • 🗄 🗎 🕘 📎 • 🗞 • 🗟 🔌 🌽 🗧	📸 • 🖄 • 🖄 • 🗞 • 📾 🔯 🖌 🗴 🗉 🗊 😭 • 😂 • 🗳 • 🐼 • 😥 • 🚱 • 🚱 • 🖉 • 🖉 • 🖉 • 🖉 • 🖉 • 🖉				
				Q Quick	Access
Project Explorer 🛿 🗖 🗖	i main.c ⊠				E 0 ⋈ "2 □ □
E 😫 😜 🗸	#include "main.h"				in 🖻 🚏 🔊 🔊 👘 🗰
▼ 😂 blinky	const wint16 t LEDS = GPIO Pin	1:			~
Binaries	const uint16_t LED[1] = {GPI0_P	in_1};			🛄 main.h
Includes	const long LED_COUNT = 1;				CLEDS : const uint16_t
includes	⊖int main()				CLED COUNT : const long
Scripts	{				<ul> <li>main() : int</li> </ul>
V 🗁 source	Toggle Breakpoint ① 米B				loop() : void
le main.c	Add Breakpoint				init() : void
In main.h	Toggle Breakpoint Enabled				delay(uint32_t) : void
startup_stars214xx.s	Breakpoint Properties				
	Breakpoint Types				
	Go to Annotation %1				
	☆ Run C/C++ Code Analysis		0		
	Add Bookmark	Properties	45	U 🔄 🔐 🔂	
	Add Task	11_CM4 -0036_310F	ENTLU-DUTLEN	-1/ USEL S/ ULIUSU	TUD/ DESKCOD/ UTILS CLUD/ WOLKS
	Add Task	H_CM4 -DUSE_STDP	PERIPH_DRIVER	-I/Users/armst	rap/Desktop/armstrap/works
	✓ Show Quick Diff ^ 企Q	H_CM4 -DUSE_STDP	PERIPH_DRIVER	-I/Users/armst	rap/Desktop/armstrap/works
	Show Line Numbers	H_CM4 -DUSE_STDP	PERIPH_DRIVER	-I/Users/armst	rap/Desktop/armstrap/works
	Folding	H_CM4 -DUSE_STDP	PERIPH_DRIVER	-I/Users/armst	rap/Desktop/armstrap/works
	Preferences	H_CM4 -DUSE_STDP	PERIPH_DRIVER -	-I/Users/armst -I/Users/armst	rap/Desktop/armstrap/works
		H_CM4 -DUSE_STOP	PERIPH_DRIVER	-I/Users/armst	rap/Desktop/armstrap/works
	Show Annotation Columns	H_CM4 -DUSE_STDP	PERIPH_DRIVER	-I/Users/armst	rap/Desktop/armstrap/works
	arm-none-eabi-acc -DSTM32F4 -DARM_M	ATH_CM4 -DUSE_STDP ATH_CM4 -DUSE_STDP	PERIPH_DRIVER ·	-1/Users/armst -I/Users/armst	rap/Desktop/armstrap/works
	arm-none-eabi-gcc -DSTM32F4 -DARM_M	ATH_CM4 -DUSE_STDP	PERIPH_DRIVER	-I/Users/armst	rap/Desktop/armstrap/works
	arm-none-eabi-gcc -DSTM32F4 -DARM_M	ATH_CM4 -DUSE_STDP	PERIPH_DRIVER	-I/Users/armst	rap/Desktop/armstrap/works
	arm-none-eabi-acc -DSTM32F4 -DARM_M	ATH_CM4 -DUSE_STDP	PERIPH_DRIVER	-I/Users/armst	rap/Desktop/armstrap/works
	arm-none-eabi-gcc -L/Users/armstrap	/Desktop/armstrap/	workspace/blin	nky/scripts -T	stm32_flash.ld -nostartfil
	13:07:12 Ruild Einished (took 4s 96	Qmc			
	15.0.112 Satta Thished (Cook 43.50				
	Writable	Smart Insert	9:2		

9. Verify the breakpoint is set by visually inspecting a blue dot in the margin.

● ○ ○ C/C++ - blinky/source/main.c - Eclipse - /Users/armstrap/Desktop/armstrap/workspace						
📑 - 🗔 🕤 👜 📎 - 🗞 - 🗟 🔪 🌛	📑 • 🗉 🖄 • 🗞 • 📾 🗙 🕖 💿 🗉 😰 • 🏜 • 📽 • 🗭 • 🚱 • 🚱 • 🚱 • 🚱 • 🚱 • 🚱 • 🚱					
					Q Quick A	Access
Project Explorer 🛿 🗖 🗖	I 📝 main.c ⊠					₽0x "2 □ □
<ul> <li>➡ Solution</li> <li>➡ Binaries</li> <li>➡ Binaries</li> <li>➡ Debug</li> <li>\_ De</li></ul>	<pre>#include "main.h" const uint16_t LED5 const uint16_t LED5 const long LED_COUN o int main() init(); doi { loop(); } while (1); return 0; // ne</pre>	= GPIO_Pin_1; 1] = {GPIO_Pin_1 T = 1; ver reached	};			P □ J <sup>4</sup> <sub>Z</sub> X X S • H I main.h CLEDS: const uintl6_t CLED: const uintl6_t[] CLED_COUNT: const long main0: int loop0: void init0: void delay(uint32_t): void
	Problems ② Tasks CDT Build Console [blinky] arm-none-eabi-gcc - DSTM arm-none-eabi-gcc -	Console 33 Pro- 2574 - DARM_MATH_ 3274 - DARM_MAT	Operaties           CM4         -DUSE_SIDP           Ktop/armstrap/	VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER VERIPH_DRIVER	v v v v v v v v v v v v v v v v v v v	Tap/Desktop/armstrap/works rap/Desktop/armstrap/works /startup_stm32f4xx.S rap/Desktop/armstrap/works /startup_stm32f4xx.S rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works rap/Desktop/armstrap/works
		Writable	Smart Insert	9:2		
k						

10. Click the debug toolbar and select your debug configuration to start flashing and debugging your Armstrap

board.

● O O C/	C++ - blinky/source/main.c - Eclines - / Users/armstrap/Desicton/arms	strap/workspace
	□ Π : @+ @+ @ + C + Q + Q + Q + Q + Q + Q + Q + Q + Q	
Image: Project Explorer S       Image: Project Explorer S         Image: Project Explorer Explorer S       Image: Project Explorer Explorer S         Image: Project Explorer Exp	<pre> @ main.c &amp;     Debug Configurations Organize Favorites const uint16_t LEDS = UFU_Prn_1; const long LED_COUNT = 1;  @ int main() {</pre>	■ O 23 *2 C L ■ J <sup>1</sup> <sub>2</sub> X X • <b>H</b> • LD : const uint16_t • LD : const uint16_t] • LD_COUNT : const long d• main0 : int • loop0 : void • int0 : void

11. If this is the first time you are debugging, you may be presented with a confirmation dialog to confirm the perspective switch. Check the **Remember my decision** option and click the **OK** button.



12. By default, Eclipse will halt on the first line of code, usually the *Reset\_Handler*. Click on the **F8** key or the green **Play** button to continue.

00	Debug	g – Source not found. – Eclipse – /Use	ers/armstrap/Des	ktop/armstrap/worksp	ace	K.
📑 - E E E 🔒	0 🛛 👷 🔳 🖉 🥆 🖓 .	e id 🤜 II 🔌 🕸 • 🖉 • 🂁 /	<b>% -</b> 1 ½ - 전 - ←	$\langle - \cdot - \cdot \rangle =   \ge 0$ Quick	Access	参 Debug
state of the second sec	Suspend	💥 🕅 🖬 🎽 🗖 🗖	(x)= Variables 🖾	©⊜ Breakpoints 🚻 Reg	isters 🛋 Modules	- 0
🔻 💽 blinky (Flash ar	nd Debug) (GDB Hardware D	Debugging]			約 🕫 🖻 🌮 🗙 📜	1 🗹 🔽
CDB Hardwa	re Debugger (10/18/13, 1:	12 PM) (Suspended)	Name		Value	
JUsers/arms	strap/Desktop/armstrap/go	c-arm/bin/arm-none-eabi-gdb (10/18/1				
/Users/arms	strap/Desktop/armstrap/we	orkspace/blinky/Debug/blinky.elf (10/18/				
c blinky (Flash ar	nd Debug) (GDB Hardware D are Debugger (10/100-2-1)	Debugging] (13 PM) (Suspended)				
Thread []	L] (Suspended)	s in (suspended)				
= 1 Rese	t_Handler() 0x080018f8	this form and with oth (10/18/1				
/Users/arms	strap/Desktop/armstrap/de	prkspace/blinky/Debug/blinky.elf (10/18/1				
c main.c C Re	eset_Handler() 🖾				🗄 Outline 🛛 👘	
No source available	for "Reset_Handler() "				An outline is not available.	
View Disassembly.						
	nata 🔍 Partitiona 🔿 I	and the second second				
	iasks 🔊 Problems 🕖 I	Executables I Memory	where a statistic statistic	un (blinder olf (10 (10 (17)		
Dinky (Hash and Deb	ug) (GDB Hardware Debugg	ingj/Users/armstrap/Desktop/armstrap/wo	orkspace/blinky/Deb	ug/biinky.eir (10/18/13, 1	(13 PM)	

13. The execution should stop at your breakpoint (as seen below) and you should be able to debug your target.

\varTheta 🔿 🔿 Debug - blinky/source/main.c - Eclipse - J	Users/armstrap/Desktop/armstrap/workspace			
11+日雨	▙ਚੋੴ© ∅ ∜ਚੋ≥ ੋ≠ ੋ≠ ← ← → +   ⊴			
	Quick Access			
券 Debug 않 🙀 🙀 🖬 😨 🗖 🗖	🕪= Variables 😫 💁 Breakpoints 👯 Registers 🛋 Modules 🗖 🗖			
▼ c blinky (Flash and Debug) [GDB Hardware Debugging]	創 皓 📄 🖉 💥 🛅 🖻 🍸			
w Thread [0] (Running)	Name Value			
// Users/armstrap/Desktop/armstrap/gcc-arm/bin/arm-none-eabi-gdb (10/18/1 // Users/armstrap/Desktop/armstrap/workspace/blinky/Debug/blinky.elf (10/18/1)				
▼ C blinky (Flash and Debug) [GDB Hardware Debugging]				
$\mathbb{V}_{II} \otimes \mathbb{C}$ GDB hardware bedugger (10/18/13, 1:15 PM) (suspended) $\mathbb{V}_{II} \otimes \mathbb{C}$ Thread [1] (Suspended: Breakpoint hit.)				
1 main() main.c:9 0x08000534 //users/armstrap/Desktop/armstrap/gcc-arm/bin/arm-none-eabi-odb (10/18/)				
. [] /I leare /armetran /Dackton /armetran /workenara /blinbu /Dahun /blinbu alf (10/18				
🖻 main.c 🛱 💽 Reset_Handler()	" 🗖 🔚 Outline 🛛 🗖 🗖			
	🔊 🖻 🛱 🖉 🖋 🛛 🗰 🔬			
<pre>const uint16_t LEDS = GPI0_Pin_1; const uint16_t LED[1] = {GPI0_Pin_1};</pre>	□ main.h ● <sup>c</sup> LEDS : const uint16 t			
const long LED_COUNT = 1;	CLED : const uint16_t[]			
⊖int main()	<ul> <li>main(): int</li> </ul>			
do init();	<ul> <li>loop(): void</li> <li>init(): void</li> </ul>			
l loop();	delay(uint32_t) : void			
$\frac{1}{2}$ while (1);				
return 0. // never reached				
Console X @ Tasks Problems O Executables 1 Memory				
blinky (Flash and Debug) [GDB Hardware Debugging] /Users/armstrap/Desktop/armstrap/	gcc-arm/bin/arm-none-eabi-gdb (10/18/13, 1:13 PM)			
monitor swdp_scan attach 1	1			
0x0800213e in ?? () set print pretty				
Note: automatically using hardware breakpoints for read-only addresse	s.			
	I			
Writable	Smart Insert 9:1			
Exploring the Dobug Teelbar				
Exploring the Debug Toolbal				
× i>   ≯   \$   ≂, Z     \$   3.   ▶ = ■	54			
The Debug toolbar includes the following buttons:				
i⇒ Instruction Stepping Mode Enables instruction stepping n	node to examine a program as it steps into disassembled code.			
Drop to Frame Re-enters the selected stack frame in the D	ebua view.			
Use Step Filters Enables step filters in the Debug view.				
- <sup>©</sup> Step Return Continues execution to the end of the current r	outine, then follows execution to the caller of the routine.			
Step Over Executes the current line, following execution in	side a routine.			

- Step Into Executes the current line, including any routines, and proceeds to the next statement.
- Resume Resumes execution of the currently suspended debug target.
- Suspend Halts execution of the currently selected thread in a debug target.
- Terminate Ends the selected debug session and/or process.
- Disconnect Detaches the debugger from the selected process.

## CHAPTER 4

## Armstrap Naming and Versioning Convension

## **Format**

Use the simple file naming convention when creating boards:

<author>\_<board-name>\_<version>.<extension>

#### <author>=([0-9][a-z]\-)\*

- lowercase-alpha-numeric string which can contain dash characters
- all uppercase characters convert to lowercase characters
- space characters convert to dash '-' characters
- invalid characters are ommited
- usually a company name, username or website domain name

#### <board-name>=([0-9][a-z]\-)\*

- · lowercase-alpha-numeric string which can contain dash characters
- all uppercase characters convert to lowercase characters
- space characters convert to dash '-' characters
- invalid characters are ommited
- usually the name of the product

#### <version>=[0-9]\*.[0-9]\*.[0-9]\*

- three numbers with a period separator
- <major-version>.<minor-version>.<micro-version>

#### <extension>=(\.brd|\.sch)

• the given extension of the PCB editor program (*brd* and *sch* are the extension for Cadsoft EAGLE files)

## **Rules**

- There are exactly two underscore '\_' characters (aka delimeters) in the filename
- author must changes when a board forks from one owner/company to another owner/company
- board-name usually never changes but can change at time of forking to preserve original 'author'
- version must change when a board is submitted to manufacturing
- version does not change when author changes on a newly forked board
- board-name and version must exist on the board silkscreen layer (top or bottom).
- [optional but highly recommended] author must exist on the board silkscreen layer (top or bottom)
- major-version is incremented when:
  - the size (dimensions) of the board changes
  - the interface (connectors) to the board change
- minor-version is incremented when:
  - any SMT chip is added
  - any SMT chip is removed
  - any SMT has moved
- micro-version is incremented when:
  - the silkscreen layer is modified
  - a trace is modified
- · major-version changing resets minor-version and micro-version to zero
- · minor-version changing resets micro-version to zero

## Remarks

- major-version changing usually requires a solder-wave/selective-solder re-tooling, re-stenciling and is the most expensive change.
- minor-version changing usually requires no solder-wave/selective-solder re-tooling, but requires re-stenciling.
- **micro-version** changing usually requires no solder-wave/selective-solder re-tooling, no re-stenciling and is generally the cheapest change

## **Workflow Example**

The Armstrap Eagle Board:

armstrap\_eagle\_1.0.0.brd

Note: armstrap-org\_eagle\_1.0.0.brd is also acceptible

The company "VOV Technology" forks the board, keeps armstrap branding but adds its own company logo:

vovtech\_armstrap-eagle\_1.0.0.brd

Note: vovtech-com\_armstrap-eagle\_1.0.0.brd is also acceptible

The company "VOV Technology" later changes changes a chip on the board but maintains the same board size and interface:

vovtech\_armstrap-eagle\_1.1.0.brd

The company "VOV Technology" later discovers a silkscreen naming problem and makes a minor change to the silkscreen layer:

```
vovtech_armstrap-eagle_1.1.1.brd
```

The Armstrap community intergrates VOV's changes, removed the VOV branding:

```
armstrap_eagle_1.1.0.brd
```

Community member 'John Smith' forks the Armstrap Eagle board into his own source code repository:

john-smith\_armstrap-eagle\_1.1.0.brd

## **Naming Examples**

#### • Valid Board Names:

```
adafruit-com_mintyboost_3.0.0.brd
sparkfun_weather-shield_1.0.0.brd
netduino-com_netduino_plus_2.0.1.brd
```

#### • Invalid Board Names:

```
mark's_arduino-motor-sheild_2.1.0.brd // author contains invalid_

→apostrophe character

arduino-bluetooth-module_1.0.0.brd // does not have exactly two_

→underscore '_' characters

supermechanical_twine_1.0.brd // missing micro-version

SparkFun.com_Current-Sensor-Breakout_2.1.0.brd // author and board name must be_

→lowercase, invalid '.' character in author
```

# CHAPTER 5

Indices and tables

- genindex
- modindex
- search