# Using Open Source Tools for STM32 Cross Development under Ubuntu Linux.

Revision 1

Author: Michel Catudal <u>mcatudal@comcast.net</u> February 14<sup>th</sup> 2009

Page 1 of 42

## Introduction

For those on a limited budget, use of open source tools to develop embedded software for the ARM devices is the only way. The goal of this document is to help those interested in doing development with Linux by making their installation of an **ARM** development system a bit easier. So far most of the information for ARM development is for windows while the Linux users are ignored. For development on Linux you are usually on your own.

The goal of this document is not to do everything for you but to make your life easier. You still need to be somewhat familiar with your Linux environment to use this document. This document describes an installation on **ubuntu 8.10 Linux.** The favorite development environment appears to be eclipse. Ubuntu 8.10 uses an old version of Eclipse. The latest release of eclipse is 3.4 and the latest development code is 3.5. This document will use version 3.5, either versions 3.3 or 3.4 could be installed. I do not provide plugins for 3.2 and have no plan to do so. I have plugins for the latest 3 versions.

For the debugging part you will need the Zylin plugin because even the latest CDT has some serious bugs in regard to debugging embedded ARM7 or Cortex devices. The Zylin source had to be patched because of the missing simulator would keep it from being compiled. I have removed the useless windows stuff.

For development you will need a set of tools. The package that I provide fully supports ARM7TDMI and cortex-m3 devices such as STM32 and Luminary.

For this demo I use a board from Olimex designed for IAR called **STM32-SK**.

If you need to rebuild the packages you need to note that in order to compile GCC you need to have it installed already as well as newlib. A work around would be to create a bootstrap version first. This is a bit more complicated.

The binaries were compiled under the 32 bits version of ubuntu 8.10. They will work on any recent debian based system.

## Hardware Information



As a hardware platform to exercise our ARM cross development tool chain, we will be using the **STM32-SK** evaluation board from Olimex at the cost of around \$250.00. You can purchase this board at <u>digikey</u>, <u>future</u> or some other distributor of STMicros. Olimex sells a slightly different board which is somewhat cheaper. You can get some cheaper boards at <u>MicroController Pros.</u>as well. Type **STM32** in the search box. You will see several interesting boards using the **STM32** processor.

The **STM32-SK** board includes two serial ports, a USB port, 1 CAN port, 16 LEDs, a LCD display, a potentiometer connected to an A/D port, a connector for a microphone and one for a headphone. It also has a connector in the back for a flash SD drive. It has standard JTAG connector for debugging.

This board comes with a limited version of the **IAR** Compiler. The Full version is very good but very expensive. The **ARM GCC** Compiler compares well against **IAR** compiler and is open source. For commercial projects you should seriously look into buying **IAR**.

With the board comes a programmer from IAR, it works nicely with windows but the **OpenOCD** support for Linux is very new and operation is rather slow compared to the competitors. This is likely to change with time.

<u>Olimex</u> has a few **USB** devices that you can buy at <u>MicroController Pros</u>



The device used with this tutorial is the **ARM-USB-TINY** which costs \$41.00. You could use something different like the **ARM-USB-JTAG** which costs \$72.00. Unless you need the extra serial port and need the power supply this device has no real advantage over the tiny device.

## **Open Source Tools**

To install an **ARM** cross development tool chain, we need the following components:

- 1. Eclipse IDE version 3.3, 3.4 or 3.5
- 2. Debugger plugin (derivative of zylin plugin)
- 3. Modified version of the gnuarm eclipse plugin.
- 4. Binutils, GNU C++/C Compiler and newlib for arm-elf targets
- 5. Latest OpenOCD for JTAG debugging

Here is a list of some of the files available at Catudal Software.

- 1. arm-elf-binutils-2.19\_020109-1\_i386.deb
- 2. binutils-cvs-020109.tar.bz2
- 3. arm-elf-gcc-4.4.0\_013109-1\_i386.deb
- 4. arm-elf-gcc-4.4.0-locales\_013109-1\_all.deb
- 5. gcc-svn-013109.tar.bz2
- 6. arm-elf-newlib-cvs\_020109-1\_i386.deb
- 7. newlib-cvs-020109.tar.bz2
- 8. arm-elf-gdb\_6.8-1\_i386.deb
- 9. arm-elf-gdb-6.8.tar.bz2
- 10. libftd2xx\_0.4.16-4\_i386.deb
- 11. openocd-0.1.0\_0.1.0-1\_i386.deb
- 12. openocd-0.1.0.tar.bz2
- 13. arm-elf-gcc-plugin-eclipse-3.3.2.tar.bz2
- 14. arm-elf-gcc-plugin-eclipse-3.4.tar.bz2
- 15. arm-elf-gcc-plugin-eclipse-3.5.tar.bz2

The locales file is only needed if you want the message to be in your native language if said language is not english. The plugin chosen will depend on which version of eclipse you use.

## Installation of C Compiler, utilities and newlib

You need to open a terminal window. You can either su root or use sudo to install the applications.

You do a cd to the directory where the files have been downloaded and install them

```
sudo debpkg -i libftd2xx_0.4.16-4_i386.deb
sudo debpkg -i openocd-0.1.0_0.1.0-1_i386.deb
sudo debpkg -i arm-elf-binutils-2.19_020109-1_i386.deb
sudo debpkg -i arm-elf-gcc-4.4.0_013109-1_i386.deb
sudo debpkg -i arm-elf-gcc-4.4.0-locales_013109-1_all.deb
sudo debpkg -i arm-elf-newlib-cvs_020109-1_i386.deb
sudo debpkg -i arm-elf-gdb_6.8-1_i386.deb
```

## Installation of Eclipse

The version of eclipse that comes with ubuntu 8.10 is too old to be usable with the latest embedded development. You need to first remove the version that is installed. You could also choose to compile your own version. It should be at least version 3.3. The latest release is 3.4.

You can download version 3.4 here. The wed address is http://www.eclipse.org/downloads/

The version we will describe here is version 3.5. This version is perfect for C or C++ development. If you prefer a more complete version use the release version 3.4. Take note that any issues found in the C C++ plugin are more likely to have been resolved in 3.5. If you choose a different version make sure that you pick the right plugins as they all have the the same names but are compressed in files with names that include the eclipse version numbers. The latest eclipse SDK as of Feb 8<sup>th</sup> is version 3.5M5. The web address is http://download.eclipse.org/eclipse/downloads/

Download and decompress the packages on the root your user account. If you have several users you may want to put it on /usr/local. The next step is to create an entry in the menu to make it easier to call it. The file to run is named eclipse, on the eclipse directory.



Start Eclipse using the menu.



The Eclipse splash screen opens up as shown here.

The first time that you run eclipse or if you have never selected a default workspace it will bring this menu. To accept the default workspace select "Use this as the default and do not ask again".

Workspace Launcher	×
Select a workspace	
Eclipse SDK stores your projects in a folder called a workspace. Choose a workspace folder to use for this session.	
Workspace: /home/michel/workspace	
☑ Use this as the default and do not ask again	
OK Cancel	





The default SDK doesn't have support for C and C++ so we need to install it.

We only need to add support for C and C++. Support for java and plugin development is already installed. Click on Next

E Inst.	all
Available Software	
Check the items that you wish to install	
Work with: All Available Sites	
	Find more software by working with the <u>Available Software Sites</u> preferences.
type filter text	
Name	Version
🗹 🖗 Eclipse C/C++ Development Platform	5.0.1.200809120802
🖌 🖗 Eclipse C/C++ Development Tools	5.0.1.200809120802
🗌 🖗 Mylyn Bridge: C/C++ Development	5.0.1.200809120802
Charting and Reporting	
Eclipse BIRT Doc	2.3.1.v20080630-2207w31191_34b
🗌 🆗 Eclipse BIRT Report Designer Framework	2.3.1.v20080630-7N7Y7AAYdliXLgz-SLsyMQ21x2fi
Collaboration Tools	
Dynamic Languages Toolkit - Mylyn Integration (Incubation)	0.95.1.v20080626-2043-08T07w31191_1402545
Mylyn Bridge: C/C++ Development	5.0.1.200809120802
Details	
Eclipse Mylyn Bridge for CDT. Allows use of Mylyn with CDT supported projects.	
	More
☑ Show only the latest versions of available software	🗹 <u>H</u> ide items that are already installed
☑ <u>G</u> roup items by category	What is <u>already installed</u> ?
0	

Click on Next to accept the installation.

6		Install		×
Install Details Review the items to be installed.				
Name	Version	Id		
Eclipse C/C++ Development Platform	5.0.1.2008091208	org.eclipse.cdt.platform.feature.group		
Eclipse C/C++ Development Tools	5.0.1.2008091208	org.eclipse.cdt.feature.group		
Size: 21 781 KB Details				
	6			< <u>III</u> >
?		< <u>B</u> ack	ext >	Cancel

It would be a good move to accept the terms. If not the install will be canceled. Click on Finish to complete the installation.

<b>G</b>	istall 🛛 🔀	
Review Licenses		
Licenses must be reviewed before the software can be installed. This include	es licenses for software required to complete the install.	
Items with licenses:	License text:	
Name Version	ECLIPSE FOUNDATION SOFTWARE USER AGREEMENT	
CDT GNU Toolchain Build Support 5.0.1.200809120802	March 17, 2005	
🖗 CDT GNU Toolchain Debug Suppor 5.0.1.200809120802	Usage Of Content	
Eclipse C/C++ Development Platfc 5.0.1.200809120802	THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE.	
Eclipse C/C++ Development Tools 5.0.1.200809120802	THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT"). USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU AGREE THAT YOUR USE OF THE CONTENT IS GOVERNED BY THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENT AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENT OR NOTICES INDICATED OR REFERENCED BELOW. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT AND THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED	
	BELOW, THEN YOU MAY NOT USE THE CONTENT.	
	I accept the terms of the license agreements	
	I do not accent the terms of the license agreements	
	< Back Next > Finish Cancel	
E In	stall	
Operation in progress		

Downloading org.eclipse.cdt.debug.mi.ui

🗆 Always r <u>u</u> n in backgroun	d		
	Run in <u>B</u> ackground	Cancel	<u>D</u> etails >>

We now close Eclipse because we need to install a plugin which will handle the project creation that we will use for the arm-elf tools. It is important to choose the correct plugin for the Eclipse version installed. All three packages have files with identical names. The plugins compiled for one version of eclipse will not work in a different version.

You need to copy the two plugins to the eclipse plugins directory.

The plugin package includes two plugins, one for ARM development and the other one a modified version of the Zylin plugin. The Zylin plugin is cleaned up of the windows and mac parts and fixed so it would actually work with Linux.

When you open eclipse again you will get an other welcome menu which you just have to remove.



Click on "Window – Other". You then get an option of which perspective you want to open. Choose C/C++.



To test our installation we will create an empty project and copy the files from a project that we know works fine.

	C/C++ - Eclipse SDK
<u>Eile E</u> dit Refac <u>t</u> or <u>N</u> avigate Se <u>a</u> rch <u>R</u> un <u>P</u> roje	ect <u>W</u> indow <u>H</u> elp
New Shift+Alt	lt+N 💙 💽 C Project
Open File <u>.</u>	🕅 C++ Project
Class	Ctrd LW/
	Ctrl I W Convert to a C/C I I Make Project
E Save	Ctrl+s
🗟 Save <u>A</u> s	
Sav <u>e</u> All Shift+	-Ctrl+S
Rever <u>t</u>	File from Template
Move	Class
Bename	F2
Refresh	F5Ctrl+N
Convert Line Delimiters To	>
<u>Print</u>	Ctrl+P
Switch Workspace	>
Restart	
import	
Export	
P <u>r</u> operties Alt-	+Enter
<u>1</u> ConfigJTAGTab.java [embeddedcdt4_0/]	
<u>2</u> plugin.xml [embeddedcdt4_0]	
<u>3</u> plugin.xml [org.eclipse.cdt.arm-elf]	
<u>4</u> CMainTab.java [embeddedcdt4_0/src/]	
Evit	📮 Console 🕱 🔲 Properties
	o display at this time.

Enter a name for the project. We will use the default location for the project. It will be a GCC ARM project. The GCC ARM project uses the arm-elf-gcc compiler.

E C Project 🗙		
C Project Create C project of selected type		
Project name: STM32-SK_LCD_Demo		
✓ Use default location		
Location: /home/michel/workspace/STM	132-SK_LCD_Demo Browse,	
Project type:	Toolchains:	
🗢 🗁 Executable	GCC ARM	
😑 Empty Project	Linux GCC	
Hello World C++ Project		
😑 Hello World ANSI C Project		
👂 🗁 Shared Library		
👂 🗁 Static Library		
👂 🗁 Makefile project		
Show project types and toolchains only if they are supported on the platform		
? < <u>B</u> ack	<u>N</u> ext > <u>Finish</u> Cancel	

We will just create a debug environment for this test so we remove the mark for the Release section. Press on next.

C Project	×
Select Configurations	
Select platforms and configurations you wish to deploy	y on
Project type: Executable	
Toolchains: GCC ARM	
Configurations:	
🖌 🔊 Debug	Select all
🔄 📣 Release	Deselect all
	Deselect all
	Advanced settings
Use "Advanced settings" button to edit project's prope	erties.
Additional configurations can be added after project cr	eation.
Use "Manage configurations" buttons either on toolbar	r or on property pages.
? < <u>Back</u> Next >	<u>Finish</u> Cancel

For now we just press on Finish. We will come back to the properties later on to fix the project configurations.

E	ete Coereb Rue Breiset Win	dow tiolo	We will import some files from a
File Edit Relactor Navig	ate search <u>Run Project win</u>	dow <u>H</u> eip	working project. We don't bring any
<u>N</u> ew Open File	Shitt+Ait+N >	~ 🏊 🗍 🥭	makefiles since the plugin will take
opennie <u>r</u>			care of creating the proper makefiles
<u>C</u> lose	Ctrl+W		needed
C <u>l</u> ose All	Shift+Ctrl+W		needed.
Save	Ctrl+S		
🗟 Save <u>A</u> s			
🐚 Sav <u>e</u> All	Shift+Ctrl+S		
Rever <u>t</u>			
Move			
Rename	F2		
🔊 Refresh	F5		
Con <u>v</u> ert Line Delimiters T	ō >		
👜 <u>P</u> rint	Ctrl+P		
Switch <u>W</u> orkspace	>		
Restart			
🚵 Import			
rt			
P <u>r</u> operties	Alt+Enter		
<u>1</u> ConfigJTAGTab.java [en	nbeddedcdt4_0/]		
2 plugin.xml [embedded			
<u>3</u> plugin.xml [org.eclipse	.cdt.arm-elf]		
<u>4</u> CMainTab.java [embed	ldedcdt4_0/src/]		
E≚it		🖳 Console 🛽	
	INO CONSOLES TO DISPLAY	at this time.	

E Import X	E Import X
Select Import resources from the local file system into an existing project.	File system         Source directory is not valid or has not been specified.
Select an import source: type filter text Comparison of the selection of	From directory: Browse
<ul> <li>▷</li></ul>	Filter Types     Select All       Into folder:     STM32-SK_LCD_Demo       Options
? < Back Next > Enrish Cancel	Create selected folders only      Cancel

3	Import from directory	
📝 < 🗟 michel 1	mp_eclipse	<u>C</u> réer un dossier
<u>R</u> accourcis	Nom	✓ Modifié
🔍 Rechercher	startup.c	Hier à 20:13
Récemment utilisés	stm32f10x adc.c	Hier à 20:13
🗟 michel	stm32f10x_adc.h	Hier à 20:13
🔳 Bureau	stm32f10x_conf.h	Hier à 20:13
🔜 Système de fichiers	stm32f10x_flash.c	Hier à 20:13
💻 Média 98,7 Mio	stm32f10x_flash.h	Hier à 20:13
🚍 Média 395,9 Gio	stm32f10x_gpio.c	Hier à 20:13
🔔 Média 300,0 Gio	stm32f10x_gpio.h	Hier à 20:13
🔔 Média 98,7 Mio	stm32f10x_it.c	Hier à 20:13
🔔 Média 57,5 Mio	stm32f10x_it.h	Hier à 20:13
🔔 /mandrivaHome	stm32f10x_lib.c	Hier à 20:13
🔔 /boot	stm32f10x_lib.h	Hier à 20:13
_/	stm32f10x_map.h	Hier à 20:13 😑
🔔 /suse	stm32f10x_nvic.c	Hier à 20:13
🔔 Média 57,5 Mio	stm32f10x_nvic.h	Hier à 20:13
- FREEDOS	stm32f10x_rcc.c	Hier à 20:13
🔜 Lecteur CD-RW/DVD:	stm32f10x_rcc.h	Hier à 20:13
🔔 /home	stm32f10x_tim1.c	Hier à 20:13
🔔 /fc8	stm32f10x_tim1.h	Hier à 20:13
	stm32f10x_type.h	Hier à 20:13
	STM32-SK-FLASH.Id	Hier à 20:13
🕂 Ajouter 📃 — Enleve	📄 syscalls.c	Hier à 20:13 🗸
	Select a directory to import from.	
		🔀 A <u>n</u> nuler 🗸 🗸 Valider

We pick all the files on the root directory of ~/temp\_eclipse/STM32-SK\_LCD\_Demo as well as the LCD directory. All the files will be copied to the ~/workspace/STM32-SK\_LCD\_Demo directory. Do not mark the STM32-SK\_LCD\_Demo directory.

E Imp	ort 🗙
File system Import resources from the local file system.	
From directory: /home/michel/temp_eclipse	Browse
✓ ☐ ➢ temp_eclipse ▷ ☑ ➢ LCD	<ul> <li>cortexm3_macro.h</li> <li>includes.h</li> <li>main.c</li> <li>openocd_stm32_jlink.cfg</li> <li>openocd_stm32_OlimexTiny.cfg</li> <li>startup.c</li> <li>STM32-SK-FLASH.ld</li> </ul>
Filter Types       Select All       Deselect All         Into folder:       STM32-SK_LCD_Demo         Options	ng
? <u>B</u> ack	Next > <u>Finish</u> Cancel

Press finish when all the files have been selected

Edit Befactor Navigate	Search Bun Project Win	dow Help	The project is almost ready. Before
<u>N</u> ew	Shift+Alt+N >		we try to compile though we need
Open File <u>.</u>			to adjust the properties.
<u>C</u> lose	Ctrl+W		
C <u>l</u> ose All	Shift+Ctrl+W		Right click on STM32-
Save	Ctrl+S		SK_LCD_Demo. On the bottom click on Properties
Save <u>A</u> s			on the bottom click on Hoperties.
蘭 Sav <u>e</u> All	Shift+Ctrl+S		
Rever <u>t</u>			
Mo <u>v</u> e			
Rena <u>m</u> e	F2		
🐑 Re <u>f</u> resh	F5		
Con⊻ert Line Delimiters To	>		
🖹 <u>P</u> rint	Ctrl+P		
Switch <u>W</u> orkspace	>		
Restart			
궡 Import			
🖾 Exp <u>o</u> rt			
P <u>r</u> operties	Alt+Enter		
<u>1</u> ConfigJTAGTab.java [embe	ddedcdt4_0/]		
2 plugin.xml [embeddedcdt	4_0]		
<u>3</u> plugin.xml [org.eclipse.cd	t.arm-elf]		
<u>4</u> CMainTab.java [embedded	dcdt4_0/src/]		
Exit		🖳 Consol	
▶ h stm32f10x_tim1.h	C-BUIIO [STM32-SK_LCI	D_Demo]	



E		Properties for STM32-5K_LCD_Demo
type filter text	Settings	<b>⇔</b> ↔ →
type filter text Resource Builders ♥ C/C++ Build Build Variables Discovery Options Environment Settings Tool Chain Editor ▷ C/C++ General Project References Run/Debug Settings	Settings	<pre></pre>
		Restore Defaults Apply
(?)		OK Cancel

We first start with the Symbols. We need to add a definition. Click on the add button.

E		Properties for STM32-SK_LCD_Demo	×
type filter text	Settings		< ⇒<
Resource			
₩ <sub>Builders</sub>	Configuration: Debug		✓ Manage Configurations
▼ C/C++ Build			
Build Variables	Tool Settings Puild Steps	Build Artifact Binany Parsers O Error Parsers	
Discovery Options	green oottinigo y Build Scope		
Settings	<ul> <li>Assembler</li> </ul>	Defined symbols (-D)	🛃 📾 兄 🖓
Tool Chain Editor	🖄 Target	thumb2	
▷ C/C++ General	Coporal (		
Project References			
Run/Debug Settings	A Target		
	Symbols		
	🖄 Directories	Undefined symbols (-U)	· · · · · · · · · · · · · · · · · · ·
	🖄 Optimization		
	🖄 Debugging		
	🖉 Warnings		
	Miscellaneous		
	General		
	A Libraries		
	Miscellaneous		
	🗢 🛞 ObjCopyFlash		
	🖄 Output		
	🖄 Section		
	🖄 Miscellaneous		
< III >			Restore <u>D</u> efaults <u>Apply</u>
?			OK Cancel

To enter the definition on new and type \_\_thumb2\_\_ and press on OK.

E	Edit Dialog	×
Defined symbols (-D)		
thumb?		
	OK Cancel	

We need to identify the linker script. Press on Browse and pick the file STM32-SK-FLASH.Id on the project directory.

		Properties for STM32-SK_LCD_Demo	
ype filter text	Settings		
Resource Builders	Configuration: Debug		Manage Configurations
r C/C++ Build			
Build Variables	Tool Settings Puild Steps Build		
Discovery Options Environment	Groot Sectings Plana Steps 2 Band		
Settings	Assembler Script (-T	/home/michel/workspace/STM32-SK_LCD_Demo/STM32-SK-FLASH.ld	<u>B</u> rowse
Tool Chain Editor	≥ Tel get ☑ Do no ≥ Debugging	t use standard start files (-nostartfiles)	
C/C++ General	🖉 General	t use default libraries (-noderaultilbs)	
Run/Debug Settings	C Compiler	arch braries (-static)	
	🖉 Symbols		
	🖉 Directories		
	Optimization		
	🖉 Debugging		
	A Miscellaneous		
	▼ 🛞 C Linker		
	🖉 General		
	🖉 Libraries		
	🖉 Output		
	2 Section		
	Aliscellaneous		
			Bestore Defaults Apply
?			OK Cancel
		×	
📄 < 🗟 michel 🛛	orkspace STM32-SK_LCD_Demo		
Baccourcis	Nom	✓ Modifié A	
Rechercher	openocd stm32 OlimexTiny.cfg	11:06	
8 Récemment utilisés	📄 startup.c	11:06	
🗟 michel	STM32-SK-FLASH.ld	11:06	
🖼 Bureau	stm32f10x_adc.c	11:06	
Systeme de fichiers	stm32f10x_adc.h	11:06	
🔔 /home	stm32f10x_flash.c	11:06	
🔔 /fc8	stm32f10x_flash.h	11:06	
🔜 Média 98,7 Mio	stm32f10x_gpio.c	11:06	
Média 395,9 Gio	stm32f10x_gpio.n	11:06	
Média 98,7 Mio	stm32f10x_it.h	11:06	
🚐 Média 57,5 Mio	stm32f10x_lib.c	11:06	
🚐 /mandrivaHome	stm32f10x_lib.h	11:06	
🔔 /boot	stm32f10x_map.h	11:06	
Jsuse	stm32f10x_nvic.h	11:00	
🔲 Média 57,5 Mio	stm32f10x_rcc.c	11:06	
FREEDOS	stm32f10x_rcc.h	11:06	
	stm32f10x_tim1.c	11:06	
수 Ajouter - Enlever	stm32f10x_type.h	11:00	
		SAnnuler ∠alider	

Page 23 of 42

The next step will be to set the library paths and libraries used. Click on add in the libraries section and enter c and gcc.

6		Properties for STM32-SK_LCD_Demo	×
type filter text	Settings		
Resource			
Builders	Configuration: Debug		Manage Configurations
▼ C/C++ Build			
Build ∨ariables	What has the states		
Discovery Options	Build Step	s We Build Artifact Binary Parsers 6 Error Parsers	
Environment	🔻 🛞 Assembler	Libraries (-I)	🗿 🝙 🖓 🐓
Settings	🖄 Target	c	
N C/C++ Coporal	🖄 Debugging	gcc	
Project References	🖉 General		
Run/Debua Settinas	C Compiler		
	Symbols		
	Directories		
	Optimization	Library search path (-L)	
	🖉 Debugging	/usr/lib/acc/arm-elf/4_4_0/thumb/thumb2	
	🖄 Warnings		
	🖄 Miscellaneous		
	🗢 🛞 C Linker		
	🖄 General		
	Dibraries		
	Miscellaneous		
	Objeopyriash     Output		
	A Section		
	Miscellaneous		
< III <b>X</b>			Restore Defaults Apply
?			OK Cancel

Click on add in the Library search path section. Enter the path for the libgcc library for thumb2 and click on OK.

Edit directory path	×
Directory:	
/usr/lib/gcc/arm-elf/4.4.0/thumb/thumb2	]
OK Cancel Workspace File system	]

Click again on add and enter the patch for the library for libc for thumb2 and click on OK.

Edit directory path	×
Directory:	
/usr/arm-elf/lib/thumb/thumb2	
OK Cancel Workspace File system.	

Next we need to setup the include paths

E		Properties for STM32-SK_LCD_Demo	×
type filter text	Settings		
Resource			
Builders	Configuration: Debug		✓ Manage Configurations
∽™C/C++ Build			
Build Variables	Tool Settings Puild Stops	Duild Artifact Ripapy Parents & Error Parents	
Discovery Options	Gibbi Settings - Bulu Steps		
Environment	🔻 🛞 Assembler	Include path (-I)	
Settings	🖄 Target	"\${workspace loc:/STM32-SK LCD Demo/LCD}"	
lool Chain Editor	🖄 Debugging	"\${workspace loc:/STM32-SK LCD Demo}"	
Project References	🖄 General	/usr/lib/gcc/arm-elf/4.4.0/include	
Bun/Debug Settings	C Compiler	/usr/arm-elf/include	
hanybebag settings	🖉 Target		
	Symbols		
	Ontimization		
	Warnings		
	Miscellaneous		
	V 🛞 C Linker		
	🖄 General		
	🖄 Libraries		
	🖄 Miscellaneous		
	🗢 🛞 ObjCopyFlash		
	🖄 Output		
	🖄 Section		
	🖄 Miscellaneous		
			Restore Defaults Apply
(V)			OK Cancel

Click on the add button and enter the local path the LCD directory.

Edit directory path	×
Directory:	
	_
"\${workspace_loc:/STM32-SK_LCD_Demo/LCD}"	
OK Cancel Workspace File system	

Click on the add button and enter the local path the project directory.

Edit directory path	×
Directory:	
"#fworkspace loc:(STM22-SK LCD Demo]"	
\${workshace_oc./stmsz-sk_ccb_benno}	_
OK Cancel Workspace File system.	
	_

Click on the add button and enter the include path for libc.

E	Edit directory path	×
Directory:		
/usr/arm-elf/include		
ОК	Cancel Workspace File system.	

Click on the add button and enter the include path for libgcc.

Edit directory path	
Directory:	
/usr/lib/gcc/arm-elf/4.4.0/include	
OK Cancel Workspace File system.	

We need to add a few options. The flags -std=gnu99 is added to support some new features. The flag --freestanding tells gcc that it is an embedded application where main doesn't require a return value. The flags -msoft-float and -mfpu=vfp are needed to make sure that the proper style of floating point is picked. It won't link if there is a mismatch between the libraries and the application code. When newlib or libgcc are not used and there are no floating point in the application the flags are not needed. If you use printf there will be a problem without those flags. When you use printf the floating point library is linked in as requested by newlib.

Properties for STM32-SK_LCD_Demo 🛛 🕅		
type filter text	Settings	
Resource Builders ▽ C/C++ Build	Configuration: Debug	Manage Configurations
Build Variables Discovery Options	Tool Settings Puild Steps Build Artifact Binary Parsers O Error Parsers	
<ul> <li>► Environment</li> <li>Settings</li> <li>Tool Chain Editor</li> <li>C/C++ General</li> <li>Project References</li> <li>Run/Debug Settings</li> </ul>	<ul> <li>▼ S Assembler</li> <li>Other flags</li> <li>-c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp</li> <li>Pobugging</li> <li>Pobugging</li> <li>Pobugging</li> <li>Potimization</li> <li></li></ul>	
( m )	Miscellaneous	Restore Defaults Apply
(?		OK Cancel

Press on apply to finish the properties settings. Now it is time to do a build. Click on the pulldown menu Project and then Rebuild Project. It should compile without error. Double check on the console at the bottom. You should see the following if everything compiled correctly.

\*\*\*\* Build of configuration Debug for project STM32-SK LCD Demo \*\*\*\* make -k all Building file: ../main.c Invoking: C Compiler arm-elf\_gcc -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -I"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/ michel/workspace/STM32-SK LCD Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/arm-elf/include -00 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"main.o" "../main.c" && \

echo -n 'main.d' ./ > 'main.d' &&  $\$ 

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -I"/home/michel/workspace/STM32-SK LCD Demo/LCD" -I"/home/michel/workspace/STM32-SK LCD Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../main.c" >> 'main.d' Finished building: ../main.c

Building file: ../startup.c

Invoking: C Compiler

arm-elf-gcc -mthumb -mcpu=cortex-m3 -D thumb2 -I"/home/michel/workspace/STM32-SK LCD Demo/LCD" -I"/home/ michel/workspace/STM32-SK LCD Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/arm-elf/include -00 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"startup.o" "../startup.c" && \

echo -n 'startup.d' ./ > 'startup.d' && \ arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -l"/home/michel/workspace/STM32-SK LCD Demo/LCD" -I"/home/michel/workspace/STM32-SK LCD Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../startup.c" >> 'startup.d' Finished building: ../startup.c

Building file: ../stm32f10x adc.c

Invoking: C Compiler

arm-elf-gcc -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -I"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/ michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/arm-elf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"stm32f10x adc.o" "../stm32f10x adc.c" && \

echo -n 'stm32f10x\_adc.d' ./ > 'stm32f10x\_adc.d' && \

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -I"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../stm32f10x adc.c" >> 'stm32f10x adc.d' Finished building: ../stm32f10x\_adc.c

Building file: ../stm32f10x flash.c

Invoking: C Compiler

arm-elf-gcc -mthumb -mcpu=cortex-m3 -D thumb2 -I"/home/michel/workspace/STM32-SK LCD Demo/LCD" -I"/home/ michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -l/usr/arm-elf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"stm32f10x\_flash.o" "../stm32f10x\_flash.c" && \

echo -n 'stm32f10x\_flash.d' ./ > 'stm32f10x\_flash.d' &&  $\$ 

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D thumb2 -I"/home/michel/workspace/STM32-SK LCD Demo/LCD" -I"/home/michel/workspace/STM32-SK LCD Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/armelf/include -00 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../stm32f10x\_flash.c" >> 'stm32f10x flash.d'

Finished building: ../stm32f10x flash.c

Building file: ../stm32f10x gpio.c

Invoking: C Compiler

arm-elf-gcc -mthumb -mcpu=cortex-m3 -D thumb2 -l"/home/michel/workspace/STM32-SK LCD Demo/LCD" -l"/home/ michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/arm-elf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"stm32f10x\_gpio.o" "../stm32f10x\_gpio.c" && \

echo -n 'stm32f10x gpio.d' ./ > 'stm32f10x gpio.d' && \

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D thumb2 -I"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../stm32f10x gpio.c" >> 'stm32f10x\_gpio.d' Finished building: ../stm32f10x\_gpio.c Building file: ../stm32f10x it.c Invoking: C Compiler

arm-elf-gcc -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -l"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -l"/home/ michel/workspace/STM32-SK\_LCD\_Demo" -l/usr/lib/gcc/arm-elf/4.4.0/include -l/usr/arm-elf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"stm32f10x\_it.o" "../stm32f10x\_it.c" && \

echo -n 'stm32f10x\_it.d' ./ > 'stm32f10x\_it.d' && \

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D\_thumb2\_\_ -I"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../stm32f10x\_it.c" >> 'stm32f10x\_it.d' Finished building: ../stm32f10x\_it.c

Building file: ../stm32f10x\_lib.c

Invoking: C Compiler

arm-elf-gcc -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -l"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -l"/home/ michel/workspace/STM32-SK\_LCD\_Demo" -l/usr/lib/gcc/arm-elf/4.4.0/include -l/usr/arm-elf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"stm32f10x\_lib.o" "../stm32f10x\_lib.c" && \

echo -n 'stm32f10x\_lib.d' ./ > 'stm32f10x\_lib.d' && \

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -l"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../stm32f10x\_lib.c" >> 'stm32f10x\_lib.d' Finished building: ../stm32f10x\_lib.c

Building file: ../stm32f10x\_nvic.c

Invoking: C Compiler

arm-elf-gcc -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -I"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/ michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/arm-elf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"stm32f10x\_nvic.o" "../stm32f10x\_nvic.c" && \

echo -n 'stm32f10x\_nvic.d' ./ > 'stm32f10x\_nvic.d' && \

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -l"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -l"/home/michel/workspace/STM32-SK\_LCD\_Demo" -l/usr/lib/gcc/arm-elf/4.4.0/include -l/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../stm32f10x\_nvic.c" >> 'stm32f10x\_nvic.d'

Finished building: ../stm32f10x\_nvic.c

Building file: ../stm32f10x\_rcc.c

Invoking: C Compiler

arm-elf-gcc -mthumb -mcpu=cortex-m3 -D\_thumb2\_\_ -I"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/ michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/arm-elf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"stm32f10x\_rcc.o" "../stm32f10x\_rcc.c" && \

echo -n 'stm32f10x\_rcc.d' ./ > 'stm32f10x\_rcc.d' && \

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D\_thumb2\_\_\_-l"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -l"/home/michel/workspace/STM32-SK\_LCD\_Demo" -l/usr/lib/gcc/arm-elf/4.4.0/include -l/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../stm32f10x\_rcc.c" >> 'stm32f10x\_rcc.d' Finished building: ../stm32f10x\_rcc.c

Building file: ../stm32f10x\_tim1.c

Invoking: C Compiler

arm-elf-gcc -mthumb -mcpu=cortex-m3 -D\_thumb2\_\_-l"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -l"/home/ michel/workspace/STM32-SK\_LCD\_Demo" -l/usr/lib/gcc/arm-elf/4.4.0/include -l/usr/arm-elf/include -00 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"stm32f10x\_tim1.o" "../stm32f10x\_tim1.c" && \

echo -n 'stm32f10x\_tim1.d' ./ > 'stm32f10x\_tim1.d' && \

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -l"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../stm32f10x\_tim1.c" >> 'stm32f10x\_tim1.d'

Finished building: ../stm32f10x\_tim1.c

Building file: ../syscalls.c

Invoking: C Compiler

arm-elf-gcc -mthumb -mcpu=cortex-m3 -D\_thumb2\_\_ -I"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/ michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/arm-elf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"syscalls.o" "../syscalls.c" && \

echo -n 'syscalls.d' ./ > 'syscalls.d' && \

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -l"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -l"/home/michel/workspace/STM32-SK\_LCD\_Demo" -l/usr/lib/gcc/arm-elf/4.4.0/include -l/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../syscalls.c" >> 'syscalls.d' Finished building: ../syscalls.c

Building file: ../LCD/drv\_hd44780.c Invoking: C Compiler arm-elf-gcc -mthumb -mcpu=cortex-m3 -D\_thumb2\_\_ -l"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -l"/home/ michel/workspace/STM32-SK\_LCD\_Demo" -l/usr/lib/gcc/arm-elf/4.4.0/include -l/usr/arm-elf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"LCD/drv\_hd44780.o" "../LCD/drv\_hd44780.c" && \ echo -n 'LCD/drv\_hd44780.d' LCD/ > 'LCD/drv\_hd44780.d' && \ arm elf acc\_MM\_MC\_R.w mthumb mcur=cortex m3 -D\_thumb2\_\_ l"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD"

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -l"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -l"/home/michel/workspace/STM32-SK\_LCD\_Demo" -l/usr/lib/gcc/arm-elf/4.4.0/include -l/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../LCD/drv\_hd44780.c" >> 'LCD/drv\_hd44780.d'

Finished building: ../LCD/drv\_hd44780.c

Building file: ../LCD/drv\_hd44780\_l.c

Invoking: C Compiler

arm-elf-gcc -mthumb -mcpu=cortex-m3 -D\_thumb2\_\_ -I"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/ michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/arm-elf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp -o"LCD/drv\_hd44780\_I.o" "../LCD/drv\_hd44780\_I.c" && \ echo -n 'LCD/drv hd44780\_I.d' LCD/ > 'LCD/drv hd44780\_I.d' && \

arm-elf-gcc -MM -MG -P -w -mthumb -mcpu=cortex-m3 -D\_thumb2\_ -I"/home/michel/workspace/STM32-SK\_LCD\_Demo/LCD" -I"/home/michel/workspace/STM32-SK\_LCD\_Demo" -I/usr/lib/gcc/arm-elf/4.4.0/include -I/usr/armelf/include -O0 -g2 -Wall -c -std=gnu99 -ffreestanding -msoft-float -mfpu=vfp "../LCD/drv\_hd44780\_I.c" >> 'LCD/drv\_hd44780\_I.d'

Finished building: ../LCD/drv\_hd44780\_l.c

Building target: STM32-SK\_LCD\_Demo.elf

Invoking: C Linker

arm-elf-ld ./main.o ./startup.o ./stm32f10x\_adc.o ./stm32f10x\_flash.o ./stm32f10x\_gpio.o ./stm32f10x\_it.o ./stm32f10x\_lib.o ./stm32f10x\_nvic.o ./stm32f10x\_rcc.o ./stm32f10x\_tim1.o ./syscalls.o ./LCD/drv\_hd44780.o ./LCD/drv\_hd44780\_l.o -T"/home/michel/workspace/STM32-SK\_LCD\_Demo/STM32-SK-FLASH.ld" -nostartfiles -static -lc -lgcc -L/usr/arm-elf/lib/thumb/thumb2 -L/usr/lib/gcc/arm-elf/4.4.0/thumb/thumb2 -Map=STM32-SK\_LCD\_Demo.map -o"STM32-SK\_LCD\_Demo.elf"

Finished building target: STM32-SK\_LCD\_Demo.elf

Invoking: ObjCopyFlash arm-elf-objcopy -O ihex STM32-SK\_LCD\_Demo.elf "STM32-SK\_LCD\_Demo.hex" Finished building: STM32-SK\_LCD\_Demo.hex



Now you need to setup some way to debug. Click on Run and Debug Configurations.

E	Debug Configurations	×	
Create, manage, and run configurations			
Image: Second system       Image: Second system         Image: Secon	Name: New_configuration     Main Debugger   Project (optional):     C/C++ Application:   Search Project     Application console	Browse Browse	
Filter matched 12 of 12 it	Apply	Re <u>v</u> ert	
?	Debug	Close	

Double click on Zylin Embedded debug and type the name that you want for the debugger menu.

E Debug Configurations 🗙			
Create, manage, and run configurations			
Image: Second system       Image: Second system         Image: Second	Name: STM-SK_LCD_Demo	n Browse B <u>r</u> owse	
Filter matched 12 of 12 it	Apply	Revert	
?	Debug	Close	

Switch to the Debugger section. Remove the .gdbinit in the command line entry since we will not use it. Make sure that arm-elf-gdb is the name of the debugger.

E	Debug Configurations		
Create, manage, and run configurations			
Image: Second system       Image: Second system         Image: Secon	Name: STM-SK_LCD_Demo      Main Commands     Debugger: Embedded GDB     Advanced        Debugger Options     Main     GDB debugger:     arm-elf-gdb        Browse     GDB command file:         (Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)        GDB command set:     Standard           Verbose console mode     Use full file path to set breakpoints		
Filter matched 12 of 12 it	Apply Revert		
?	ebug Close		

E Debug Configurations			
Create, manage, and run configurations			
Image: Second system       Image: Second system         Image: Secon	Name:       STM-SK_LCD_Demo         Main Statut       Debugger       Commands         Help/tips on how to setup GDB init script         'Initialize' commands         target extended-remote localhost:3333         mon halt         load         mon cortex_m3 maskisr off         thbreak main         c         'Run' commands		
Filter matched 12 of 12 it	Apply	vert	
?	Debug	Close	

Instead of having a .gdbinit file we enter the startup commands in the "'Initialize' commands" section. The reason for that is so we can follow what is going on in the Console section and not wondering whether or not something has crashed.

The commands to enter are : target extended-remote localhost:3333 mon halt load mon cortex\_m3 maskisr off thbreak main c

mon is short for monitor. It is needed when the command is for OpenOCD and not a gdb command. The load command will program the flash with our program. Because of a bug in OpenOCD where the maskisr is not restored. This cripples the interrupt structure, making it impossible to have any interrupt when debugging.

An alternative may be this. This would turn the interrupts off when breaking and reenable them when restarting. Not fully tested, I got that from a newsgroup.

target extended-remote localhost:3333 #turn off timers and peripherals while we are stopped mon halt #mask interrupts when we stop, re-enable when we turn back on. define hook-stop mon cortex\_m3 maskisr on end define hook-continue mon cortex\_m3 maskisr off end load mon cortex\_m3 maskisr off thbreak main

С

Create, manage, and r	in configurations
Image: Second system       Image: Second system         Image: Secon	Name:       STM-SK_LCD_Demo            Main
Filter matched 12 of 12 it	Workspace       File System       Variables         □ Append       Image: Apply

×

We unmark "Launch in background" so we know when loading is done and don't assume that it is crashed, it also gives us the option to cancel if we conclude that it did crash. Mark the box near Debug so it add it to the top menu. You need to click on Apply when done and then close so you don't have to redo the whole thing.

You would probably want to create another launcher, on that doesn't include the load command for faster load when you don't need to reflash the device. The setup is almost identical to the one used to flash the device and go into debug.

The commands to enter for that launcher

target extended-remote localhost:3333 mon halt thbreak \_startup R thbreak main c

In this case the application is allready programmed and running. We have to stop it, reset the program, set a temporary breakpoint at main and restart the program. It will stop at the breakpoint. Keep in mind that only 2 breakpoints are allowed so using a fixed breakpoint for this would leave you with only one breakpoint. Any code that runs out of RAM is not bound with this limitation. When you need several breakpoints, you need to compile some of the code to run out of RAM. In that case you will have to make sure that your startup code copies the actual code in RAM otherwise this will hang. The startup file needs to be modified for that to occur.

			C/C++ - Eclipse SDK	
	<u>Bun</u> <u>P</u> roject <u>W</u> indow <u>H</u> elp	)		
\$	₿un	Ctrl+F11		~ *
-	🇞 <u>D</u> ebug	F11		
_	Run His <u>t</u> ory	>		
	Run A <u>s</u>	>		
	Ru <u>n</u> Configurations			
	Debug History	>		
	Debug As	>		
	_ De <u>b</u> ug Configurations			
	Toggle Breakpoint	Shift+Ctrl+B		
	<ul> <li>Toggle Line Breakpoint</li> </ul>			
	<ul> <li>Toggle <u>M</u>ethod Breakpoint</li> </ul>			
	% Toggle <u>W</u> atchpoint			
	🗆 S <u>k</u> ip All Breakpoints			
	💥 Remo <u>v</u> e All Breakpoints			
	💁 <u>E</u> xternal Tools	>	(no launch history)	
			Bun As >	
			External Tools Configurations	
			Organize Fa⊻orites	
			1	

arm-elf-gdb communicates with OpenOCD with target extendedremote localhost:3333. We need to create a launcher to start OpenOCD before we attempt to load the debugger. Click on "External Tools – External Tools Configurations" Highlight Program and click on new

External Tools Configurations			
Create, manage, and run configurations			
Run a program			
Image: Constraint of the second s	Name: OpenOCD		
象 Ant Build			
🗢 💁 Program	/usr/bin/openocd		
💁 OpenOCD	Browse Workspace Brows <u>e</u> File System Varjables		
	Working Directory:		
	\${workspace_loc:/STM32-SK_LCD_Demo}		
	Browse Wor <u>k</u> space Browse File Syste <u>m</u> Varia <u>b</u> les		
	Arauments:		
	-f \${workspace_loc:/STM32-SK_LCD_Demo}/openocd_stm32_OlimexTiny.cfg		
Note: Sinclose an argument containing spaces using double-quotes (").			
Filter matched 3 of 3 items	Appl <u>y</u> Re <u>v</u> ert		
?	Bun Close		

Type /usr/bin/openocd in the location. For the Working Directory click on Browse Workspace and accept the workspace where the project is. For the argument enter the configuration used for your JTAG programmer. If you use something else than jlink or Olimex Tiny you need to create your own configuration file.

Click on Apply and change to the Common page.

External Tools Configurations			
Create, manage, and run configurations			
Run a program			
Image: Second secon	Name: OpenOCD		
条 Ant Build	Save as		
▼ Q Program	● L <u>o</u> cal file		
Va OpenOCD	O Shared file: Browse		
	Display in favor <u>i</u> tes menu		
	External Tools     O Default - inherited (UTF-8)		
	Standard Input and Output		
	Workspace, File System, Variables,		
	□ Append		
	☑ Launch in background		
Filter matched 3 of 3 items	Apply Revert		
?	<u>B</u> un Close		

Mark the box for External Tools so it will appear as a menu.

	C/C++ - Eclipse SDK	New we we do to be the sec
Bun       Project       Window       Help	J € V II I	Now we need to test our setup. Click on Run – External Tools -OpenOCD. This will start OpenOCD.
External Tools	▲ 1 OpenOCD       Bun As       External Tools Configurations       Organize Fayorites	

This will appear on the bottom console

Open On-Chip Debugger 0.1.0 (2009-02-07-00:57) Release

BUGS? Read http://svn.berlios.de/svnroot/repos/openocd/trunk/BUGS

\$URL: https://kc8apf@svn.berlios.de/svnroot/repos/openocd/tags/openocd-0.1.0/src/openocd.c \$ jtag\_speed: 20 500 kHz

Info : JTAG tap: stm32.cpu tap/device found: 0x3ba00477 (Manufacturer: 0x23b, Part: 0xba00, Version: 0x3)

Info : JTAG Tap/device matched

Info : JTAG tap: stm32.bs tap/device found: 0x16410041 (Manufacturer: 0x020, Part: 0x6410, Version: 0x1)

Info : JTAG Tap/device matched

Warn : no tcl port specified, using default port 6666

Now switch the perspective to debug. You should see the following window

C Debug - Eclips	e SDK	
<u>E</u> ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>B</u> un <u>P</u> roject <u>W</u> indow <u>H</u> elp		
] 🗈 · 📓 🚔   第 · O · Q · ] 🥭 🖋 · ] 원 · 칭 · ♡ · ↔ ·		🖺 🏇 Debug 🗟 ( 🎽
拳 Debug X	🕬= Variables 🛿 🔏 Breakpoints	🏝 🐗 🖻 🖇 💥 🌄 🖻
♥ QpenOCD [Program]		()   10
	- 8	E Outline 🕱 🗖 🗖
Console X Variation of Executables		
OpenOCD (Program) /usr/bin/openocd		۲ ۲ ۲
] 0°		



Once OpenOCD is running we can start the debugger

E	Progress Information X
	Launching STM-SK_LCD_Demo
GDB co	mmands: mon halt
	Cancel

This window appears as the program is being flashed into the STM32 device. If it takes way too long press on cancel and retry again. I have found that with that STM32-SK board reflashing always fails the first time right after power up but usually succeeds on the second try. To restart a failed load right click on the debug line for the project and click on Terminate and Remove. Then click on the debug menu again. It should work this time.

If you don't have something like this in the console section on the bottom there is a problem and you will need to restart until you get it to work right. I have not seen a case yet where it doesn't work the second time after power up.

```
target extended-remote localhost:3333
mon halt
Loading section .data, size 0x50c lma 0x20000000
load
Loading section .text, size 0x7458 lma 0x8000000
Loading section .eh_frame, size 0x2648 lma 0x8007458
Loading section .rodata.str1.4, size 0x34 lma 0x8009aa0
Start address 0x8000285, load size 40928
Transfer rate: 2 KB/sec, 8185 bytes/write.
mon cortex_m3 maskisr off
cortex_m3 interrupt mask off
Hardware assisted breakpoint 1 at 0x800028a: file ../main.c, line 185.
thbreak main
C
main () at ../main.c:185
      InitADC();
185
```

You are now ready for debugging.



This document is Copyright (C) Michel Catudal 2009 All rights reserved.

If you feel that this has been helpful to you feel free to make a donation with paypal at <a href="mailto:mcast.net">mcatudal@comcast.net</a>

**Revision History** 

Release 1 : First draft