# Configuring Ubuntu to Code for the OmniFlash or OmniEP

## **Table of Contents**

Introduction
Assumptions2
Getting Started2
Getting the Cross Compiler for ARM2
Extracting the contents of the compressed file
Copying the compiler into the proper place
Configuring CodeBlocks to use the Cross Compiler
Starting a new Project in CodeBlocks
Adding a new Build Target for the ARM processor15
Testing the Compiler
Adding #defines to a project
Testing our application locally
Running our new program on the ARM Processor
Serial Port Configuration - PuTTY24
Connection Verification - PuTTy27
Serial Port Configuration - CuteCom29
Testing connection with CuteCom
Sending a program via CuteCom32
Verifying the program was received by the OmniFlash
Launching the program
Final Notes

#### Introduction

This document describes the steps for setting up Ubuntu to develop and code for JK Micro's OmniFlash and OmniEP embedded devices (See <u>http://www.jkmicro.com</u>).

#### Assumptions

It is assumed the reader has a working Ubuntu operating system. For instructions on how to set this up, please see the document entitled "Installing and configuring Ubuntu Linux.docx". It goes over in detail exactly how to get up and running in Ubuntu Linux. It is also assumed that the following packages have been installed via the Synaptic Package Manager.

- 1.) codeblocks
- 2.) codeblocks-contrib
- 3.) g++
- 4.) cutecom
- 5.) Irzsz
- 6.) Putty and/or gtkterm
- 7.) xutils-dev (if you want to build from command line and use makedepend)

### **Getting Started**

We need to download the cross compiler for the OmniFlash and OmniEP as well as configure Code::Blocks to compile our programs.

### **Getting the Cross Compiler for ARM**

We can download the cross compiler from Cirrus Logic at this website: <a href="http://arm.cirrus.com/files/index.php?path=tools/">http://arm.cirrus.com/files/index.php?path=tools/</a>

Cirrus Logic EPD ARM :: EPD Downloads - Shiretoko										
<u>File Edit View History Bookmarks Tools Help</u>	1									
← → ✓ 🗞 🛞 🏫 💿 http://arm.cirrus.com/files/index.php?path=tools/ 🏠 ✔ 💽 ✔ Google										
🛅 Most Visited ∽ 🔹 🏚 Getting Started 🔂 Latest Headlines ∽										
Cirrus Logic EPD ARM :: EPD Do					~					
CIRRUS LOGIC <sup>®</sup>										
http://arm.cirrus.com/files/tools			date	1						
name -	type	size		descript	on					
	<dik></dik>	00.2 MB	25-05-06							
+ CP arm-eif-gcc-3.2.1-tuil.tar.bz2	DZ2	88.2 MB	27-04-05							
+ CP arm-linux-gcc-3.3-ruit.tar.bz2	DZZ	110.7 MB	27-04-05							
+ CP arm-linux-gcc-3.4.3-1.0.1.tar.bz2	DZZ	189.7 MB	21-07-06							
± [] arm-linux-gcc-4.1.1-920t.tar.bz2	DZ2	29.0 MB	04-04-07							
+ D Dundroot-mavcrunch-4.1.1.Src.tar.b22	DZZ	522.0 MB	29-05-07							
± □ crunch-tools-1.4.0.tar.b22	DZZ	1.7 MB	01-10-07							
$\pm$ [] <sup>2</sup> Crunch-tools-1.4.1-2.tdf.D22	DZZ	1.7 MD	27-11-07							
$\pm$ [] Crunch-tools-1.4.3.tal.022	DZZ	33.7 MD	04.05.07							
2 Lp gcc-4.1.1-mavenek-eabilitai.bz2	DZZ	29.5 MB	04-05-07							
Done										

For the **OmniFlash**, download the "**arm-linux-gcc-3.3-full.tar.bz2**" file. For the **OmniEP**, download the file "**arm-linux-gcc-4.1.1-920t.tar.bz2**". I haven't played around too much with compiling for the OmniFlash with the 3.43 or 4.1.1 compiler yet. It may work.

This document will cover the installation steps for the OmniFlash. The steps for OmniEP are the same, just with a different compiler (the 4.1.1 version).

Save the file somewhere on your system. It defaults to your desktop. You can change this in the options of Firefox.



#### Extracting the contents of the compressed file

Navigate to where you saved your file. Right-click on it and click Extract here.

File Edit View Go Bookmarks Tabs Help   Back Forward Up Stop Reload ~   Back Forward Up Stop Reload ~   Location: Documents/Downloads Image: Comparison of the second o		ownio	oads - File B	rowse	r [	- • ×				
Back Forward Up Stop Reload   Location: Documents/Downloads I 100% I   Places Image: Comparison of the system Image: Comparison of the system Image: Comparison of the system   Image: Network File System Image: Comparison of the system	<u>File Edit View</u>	/ <u>G</u> o	<u>B</u> ookmarks	Tabs	<u>H</u> elp					
Location: Documents/Downloads Places V Desktop File System Network	🚑 🖌 🚔 Back Forwa	rd	😭 Up	Stop	C Reload	~				
Places   Desktop   File System   Network	Location: /Documents/Downloads 🔍 100% 🔍									
Flood Drive	Places > Desktop File System Network Floppy Drive		usr		arm-linux full.ta	-gcc-3.3- r.bz2				

Once extracted, you should have a folder like the one above called usr.

## Copying the compiler into the proper place

We need to copy the contents to a location where we can access it globally.



Click on Applications->Terminal to get a terminal session.



Type **sudo nautilus** to bring up a file explorer that has root access. It is important that it has root access so we can copy files to a protected folder.



From the new File Browser that just opened up, navigate to the /usr/local folder.

In the other folder where we extracted the compiler to, navigate inside the **usr** folder and then inside the **local** folder. You fill find an **arm** folder.

Copy the **arm** folder (by dragging it over) into **the /usr/local** folder.

🛟 Applications Places System 🕹 🗟 🖓		ः 🚍 👮 剩 Sun Aug 30, 11:35 PM 🛛 । भ 💷
local - File Browser		
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>G</u> o <u>B</u> ookmarks <u>T</u> abs <u>H</u> elp		
Herein Contraction     Herein Contraction       Back     Forward       Up     Stop       Reload		
	local - File Browser	
Location: hts/Downloads/usr/local	<u>B</u> ookmarks <u>T</u> abs <u>H</u> elp	
Places V Q		• •
🖾 Desktop	I 🔍 100% 🔍 Icon	View 🗸
🔜 File System		
🐺 Network		
Eloppy Drive		
"arm" selected (containing 1 item)	arm bin	
- File System		-
<u>File Edit View Terminal Help</u>		
Isudol password for	etc games	
Nautilus-Share-Message: Called "net us 🖵 floppy0	Fi Fi	le Operations
e' returned error 255: net usershare: 👼 Trash 🔍	Conving O files (in llowell) to II	le ce lu
amba/usersnares. Error No such file of 9 items, Free space: 14.	7 GB Copying 8 mes (in ann ) to 1	
** (nautilus:5836): WARNING **: Unable to add monitor: Ope	ration not	

Wait for the copy process to complete.



You should now have an arm folder under /usr/local. This is the cross compiler which we will use to set up Code::Blocks.

#### Now close all open windows.

## **Configuring CodeBlocks to use the Cross Compiler**



Launch the CodeBlocks IDE.

Co Co	mpilers auto-detection	×	
Compiler	Status	Set as default	
GNU GCC Compiler Intel C/C++ Compiler SDCC Compiler Tiny C Compiler GDC D Compiler Digital Mars D Compiler GNU ARM GCC Compiler GNU AVR GCC Compiler GNU GCC Compiler for PowerPC	Detected		
Default compiler: GNU GCC Cor	mpiler		<b>cks</b> form IDE
	8.02		org

Click OK to this window. It only shows up the first time. We will configure the ARM GCC compiler.

Start here - Code::Blocks 8.02													
<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	Sea <u>r</u> ch	<u>P</u> roject	<u>B</u> uild	<u>D</u> ebug	<u>w</u> xSmith	<u>T</u> ools	P <u>l</u> ugins	<u>S</u> ettings	<u>H</u> elp		
	<b>1</b>	a 🥑	123	1 % 6	BI	g B				<u>E</u> nviro	nment		
										E <u>d</u> itor.			
: _										( <u>C</u> ompi	ler and deb	ugger	
1	唱 ?	3	?> </p	$\Theta \mid \Box$	l.					<u>G</u> lobal	variables		
0			😥 🛛 Bui	ild target:			0			<u>s</u> cripu	ng		
							0			Edit st	artup script		
Manag	ement			×	rt harr	. <b>.</b> .	1						4 4
Pro	jects	Resou	urces 🖣	> Sta	rt nere								
Projects       Resources       Image: Code Shippets         Workspace       Image: Code Shippets       Search results       Image: Code Shippets         OpenFilesList       AStylePlugin       Running startup script       Image: Code Shippets													
												default	

Click on Settings -> Compiler and debugger...

-	Compiler and debugger se	ttings 🛛 🗙
<b>Global compiler se</b>	ettings	
$\sim$	Selected compiler	
	GNU GCC Compiler Intel C/C++ Compiler SDCC Compiler	Rename Delete Reset defaults
Global compiler settings	Tiny C Compiler GDC D Compiler	cer settings Search directories
	GNU ARM GCC Compiler	↓ 
Profiler settings	GNU GCC Compiler for PowerPC GNU GCC Compiler for TriCore	

Click the pull down menu for the compilers and choose the GNU ARM GCC Compiler

Compiler and debugger settings									
Global compiler settings									
~~~	Selected compiler GNU ARM GCC Compiler								
	Set as default	Copy Rename Delete Reset defaults							
Global compiler settings	Linker settings	Search directories Toolchain executables							
	Compiler's installation di /usr NOTE: All programs belo	rectory							
Profiler settings	this path	nal nathe"							
Š	Program Files Additiona C compiler:	al Paths arm-elf-gcc							
Batch builds	C++ compiler:	arm-elf-g++							
$\otimes$	Linker for dynamic libs:								
	Linker for static libs:								
Debugger settings	Debugger:	arm-elf-gdb							
	Resource compiler:								
	Make program:	[make							
Cancel QK									

Click the arrow to the right of the tabs until you come to the Toolchain executables. The defaults listed are wrong. We need to specify what to use.

•	Compiler and d	ebugger settings					
Global compiler se	ttings						
$\sim$	Selected compiler						
503	GNU ARM GCC Compile	r					
	Set as default	Copy Rename Delet					
Global compiler settings	<ul> <li>Linker settings</li> </ul>	Search directories To					
	Compiler's installation d	lirectory					
	/usr/local/arm/3.3						
Profiler settings	NOTE: All programs below, must exist either in the "bin" this path or in any of the "Additional paths"						
	Program Files Additional Paths						
	C compiler:	arm-linux-gcc					
Batch builds	C++ compiler:	arm-linux-g++					
$(\mathbf{x})$	Linker for dynamic libs:	arm-linux-gcc					
	Linker for static libs:	arm-linux-ar					
Debugger settings	Debugger:	arm-linux-gdb					
	Resource compiler:						
	Make program:	make					

Change the compiler's installation directory to **/usr/local/arm/3.3**. If you installed a different compiler, select it's directory MINUS the bin directory. The program appends the bin directory automatically to the path you enter here. The compiler used for dynamic libs is not a mistake. We cannot use the C++ compiler for libraries on this platform for some reason. It doesn't work (or I haven't found out how to make it work yet).

There are lots of other settings for Code Blocks. I recommend you go through each screen and set it up to your liking.

## Starting a new Project in CodeBlocks

-								Start	here - C	ode::Bloc	ks 8.02	
<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	Sea <u>r</u> ch	Project	<u>B</u> uild	<u>D</u> ebug	<u>w</u> xSmith	<u>T</u> ools	P <u>l</u> ugins	<u>S</u> ettings	<u>H</u> elp	
	<b>i</b>	8	🗶 🔖	1 2 6	<b>B</b>	9, B	:			<ul> <li>\begin{bmatrix} \lambda \\ \lambda \\</li></ul>		
		0° 7.	{*} ?</th <th>3   🗖</th> <th>1</th> <th></th> <th>• • •</th> <th>Build</th> <th>target:</th> <th></th> <th></th> <th></th>	3   🗖	1		• • •	Build	target:			
							\$					\$
Manag	gement	t	( 	Sta	rt here	×						
(	O Wo	rkspac	2	2				E.	Releas	e 8.02 (uni	Code::Block The open source, cross-platform I http://www.codeblocks.org known date) gcc 4.3.3 Linux/un	S DE icode

From the main window of CodeBlocks, click the Create a new project button.



Click the Console application and click GO.

1	Console application
	Please make a selection C++
	< <u>B</u> ack <u>Next</u> > <u>S</u> Cancel

Click the C language and Next.

1	Console application	×
🐻 Console	Please select the folder where you want the new project to be created as well as its title.	
	Project title: armtestapp Folder to create project in:	
	/home/r////////////////////////////	
	Project filename:	
	armtestapp.cbp	
	Resulting filename:	
	/home/n II / I/code/armtestapp/armtestapp.cbp	
	< <u>Back</u> <u>Next</u> >	

Give your project a name. Click the ellipses button next to the folder to create your project in and navigate to a folder you want to work in. Verify the resulting filename and click Next.

•	Console application	X
🚮 Console	use and which configurations ject.	
	GNU GCC Compiler	~
	[]	
	🗹 Create "Debug" configurat	tion: Debug
	"Debug" options	
	Output dir.: bin/D	ebug
	Objects output dir.: obj/Debug	ebug
	✓ Create "Release" configuration	ation: Release
	"Release" options	]
	Output dir.: bin/Re	elease
	Objects output dir.: obj/Re	elease
	< [	Back <u>Finish</u> SCancel

For now, accept the defaults. We will fill in the ARM compiler section later. Click Finish.

## Adding a new Build Target for the ARM processor

<b>*</b>				n
<u>F</u> ile <u>E</u> dit <u>V</u> iew Sea <u>r</u> ch	<u>P</u> roject	<u>B</u> uild	<u>D</u> ebug	<u>w</u> xSm
] ≅ ⊟ ∰   & >   ↓	Add f Add f Remo	iles iles reci ove files	ursively	
Management (Projects Resources 4	Proje Set <u>p</u> Notes	ct tree rogram: s	s' argume	ents
<ul> <li>Workspace</li> <li><b>Image: Sources</b></li> </ul>	Build Prope	options erties	 )	

Click Project -> Properties...

•				Pro
Project sett	ings	Build targets	)	Build so
Build targets	_		1	Selected
Debug		Add		Platform
Release		Rename		Туре:
		Duplicate	]	
		Delete		

Click the Build targets tab. Select the Release target and click Duplicate.

Par Dup	licate build target	×
Enter the duplic	ated build target's name:	
armRelease		
		_
	Cancel	:

Give it a name. I chose armRelease.

		Project/ta	argets	options			
Project setting	gs Build targets	Build scripts	Notes	Debugger	Libraries	C/C++ parser opti	
Build targets		Selected build	target o	options			
Debug	Debug Add		Platforms:		All		
Release	Release Rename		Туре:		Console application		
	Duplicate			Pause w	hen execut	tion ends	
	Delete			Create .	DEF export	s file	
		Output filenam	e:	bin/armRel	lease/armte	estapp	
	Virtual targets			Auto-ge	nerate filer	name prefix	
	Dependencies	Execution work	king dir:	Muto-ge	nerate nier		
	Re-order	Objects output	dir:	ob /armRe	lease/		
	Build options	Build target file	es:				
		main c					

Make sure you have the armRelease target specified from the Build targets. Now add this name to the "Output filename" and "Objects output dir" or you will overwrite your regular release objects.

Now click the Build options... button.

-		Project build options								
	armtestapp Debug Polooso	Selected compiler GNU ARM GCC Compiler								
	armRelease	Compiler settings	Linker settings	Search directories >						
		Policy: Append target opt	tions to project optior	ns 🗘						

Select the armRelease target and then change the compiler to the ARM GCC Compiler. Click **OK** to save the settings and the **OK** again.

## **Testing the Compiler**

Now let's test our new compiler out.

<b>*</b>	main.c [armtestapp] - Code::Blocks 8.0
<u>F</u> ile <u>E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>F</u>	oject <u>B</u> uild <u>D</u> ebug <u>w</u> xSmith <u>T</u> ools P <u>l</u> ugins <u>S</u> ettings <u>H</u> elp
	2 D 🛍 I 💁 🖳 📔 🔍 🔍 🔍
💵 🖷 🖗 🐍 🏷 🧭	🗖 🧞 💊 💊 🐼 🐼 (Build target:) Debug
	a main() : int
Management 🕱	main.c ×
Projects Resources	1 #include <stdio.h></stdio.h>
🗢 🚺 Workspace	2 #include <stdlib.h> 3</stdlib.h>
🗢 🛃 armtestapp	4 int main() 5 □ {
🗢 🖻 Sources	6 printf("Hello world!\n"); 7 return θ;
📄 <mark>main.c</mark>	8 <sup>L</sup> } 9

Change the Built target type to our armRelease target we just set up.



Click the Build button (or the Rebuild all button).



Check for success down in the Build Log.

Congratulations! You successfully configured the CodeBlocks editor to compile for the OmniFlash ARM processor.

#### Adding #defines to a project

Sometimes you need to add conditional #defines to your project. For example, some code statements may work on one platform, but not on others.

```
main.c×
         #include <stdio.h>
   1
   2
         #include <stdlib.h>
   3
   4
         int main()
   5
      ⊟{
   6
             int val = 0;
   7
   8
         #ifdef ARM PROCESSOR
   9
             printf("Hello ARM World!\n");
  10
             val = 1;
  11
         #else
             printf("Hello PC World!\n");
  12
  13
             val = 2;
  14
         #endif
             printf("The value of val is %d\n", val);
  15
  16
             return 0;
        L}
  17
```

Take for example the following code snippet. We need to define **ARM\_PROCESSOR** when we build for the arm.

Pro	oject	<u>B</u> uild	<u>D</u> ebug	<u>w</u> xSmith						
	Add files									
	Add files recursively									
	Remo	ve files								
	Project tree >									
	Set <u>p</u>	rograms	s' argume	ents						
	Notes	5								
C	Build	options	i							
	Properties									
	Autoversioning									
	Increment Version									
	Changes Log									

Click on Project -> Build options...

8	Project build options ×				
armtestapp Debug Release	Selected compiler GNU ARM GCC Compiler				
armRelease	Compiler settings Linker settings Search directories          Policy:       Append target options to project options         Compiler Flags       Other options         #defines				

From the left, click on the armRelease target. Then click the #defines tab and add your defines. If you have more than one define to add, add them each on separate lines.

## **Testing our application locally**

Before we run the application on the ARM processor, we can test and debug our application on the Linux system and work out the kinks.



Change the build target to Debug.

Compile the program by either clicking the Build icon or the Rebuild All icon.

To set a breakpoint, click just to the right of the line number.

-	main.c [armtestapp] - Code::Blocks 8.02
<u>F</u> ile <u>E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>P</u>	roject <u>B</u> uild <u>D</u> ebug <u>w</u> xSmith <u>T</u> ools P <u>l</u> ugins <u>S</u> ettings <u>H</u> elp
1 🖻 🖻 🗿   🔦 📎	2 0 🖺 I 💁 🖳 📔 🔍 🎴 🖉
🚺 🖷 ि 🚡 🏷 ल 😡	🗖 🧞 🚫 👂 🧔 🕢 Build target: Debug
	🗘 🗌 main() : int
Management 🕱	main.c ×
Projects       Resources       4         ▼       Workspace         ▼       ■ armtestapp         ▶       > Sources	<pre>1 #include <stdio.h> 2 #include <stdib.h> 3 4 int main() 5  = { 6 int val = 0; 7 8 #ifdef ARM_PROCESSOR 9 printf("Hello ARM World!\n"); val = 1; 11 #else 12 printf("Hello PC World!\n"); val = 2; 14 #endif 15</stdib.h></stdio.h></pre>
	16 return 0; 17 }

This picture shows setting a breakpoint, building the application, and then running it in debug.

1	main.o	: [armtestapp] - Code::Blocks 8.02	
<u>File Edit View Search F</u>	Project <u>B</u> uild	<u>D</u> ebug <u>w</u> xSmith <u>T</u> ools P <u>l</u> ugins <u>S</u> ettings <u>H</u> elp	
🗋 🖻 🗃 🎒 🔍 📎 I	2 0 🛍	9. B	
	E		Program Console
1 📲 🦛 (F 📅 (F) (F) 🚺		Solution and the second	<u></u>
		ain() : int	2
Management 🕱			
	main.c ^		
Trojects Resources	1	<pre>#include <stdio.h></stdio.h></pre>	
🗢 💽 Workspace	2	#include <stdlib.h></stdlib.h>	
	3		
	4	int main()	
Sources	5		
	6	<b>int</b> val = 0;	
	/	Wifelaf ADM DDOCECCOD	
	8	#ITGET ARM PROCESSOR	
	9	printl("Hello ARM World!(n");	
	10	Val = 1;	
	11	#else	
	12	printi("Hello PC world!(h");	
	13	Vdl = 2; #ondif	
	14	<pre>#cliuit printf("The value of val is %d\n" val);</pre>	
	15	return 0.	
	10		
	1/ -	1	

Here is our debugger output.

Det	bug	<u>w</u> xSmith	Tools	P <u>l</u> ugins	<u>S</u> ettings	<u>H</u> elp	
10	<u>S</u> ta	rt				F8	
8	Sto	p debugger					
IB	Cor	ntinue				Ctrl+F7	
8	Nex	ct line				F7	,   <b>0</b>
7	Nex	t instructio	n			Alt+F7	
?>	Ste	p into				Shift+F7	·
< <b>(</b> 5	Ste	p out			Shit	ft+Ctrl+F7	·
1	Toggle breakpoint				F5		
	Ren	nove all bre	eakpoint	ts			
ų 🙀	Run	to cursor				F4	te i i i i i i i i i i i i i i i i i i i
÷	Add	symbol fil	e				=
#	Deb	ougging wir	ndows			2	Breakpoints
	Info	ormation				2	Call stack
4	Edit	t <u>w</u> atches					CPU Registers
	Atta	ach to proce	ess				Disassembly
	Det	ach					Examine memory
1	Sen	id user com	mand to	o debugge	er		<ul> <li>Running threads</li> <li>Watches</li> </ul>

To enable the watch window and any other debugging information,

Click Debug -> Debugging windows -> Watches (or any other window). Then you can drag the window(s) and dock them into your main window somewhere.

-	main.c [armtestapp] - Code::Blocks 8.02
<u>F</u> ile <u>E</u> dit <u>V</u> iew Sea <u>r</u> ch <u>F</u>	project <u>B</u> uild <u>D</u> ebug <u>w</u> xSmith <u>T</u> ools P <u>l</u> ugins <u>S</u> ettings <u>H</u> elp
1 🖻 🗃 🗿   🔦 🦻	
🚺 🏰 😤 🖓 🚠 🔿 🗗	📃 🛄 🤹 🕨 🥵 🐼 🐼 🛛 Build target: Debug
	ain() : int
Management 🕱	main.c ×
Projects Resources <sup>↓</sup>	1 #include <stdio.h></stdio.h>
	<pre>2 #include <stdlib.h></stdlib.h></pre>
🗢 🗧 armtestapp	4 int main()
Sources	5 🖂 {
	6 int val = 0;
	8 #ifdef ARM PROCESSOR
	<pre>9 printf("Hello ARM World!\n");</pre>
	10 val = 1;
	11 #else
	val = 2;
	14 #endif
Watches 🔀	<pre>15 printf("The value of val is %d\n", val);</pre>
✓ Local variables	16 return 0;
val = 2	17 -}
✓ Function Arguments	K III
No arguments.	Logs & others
	📝 Code::Blocks 🔍 Search results 🔍 Thread search 🤤 Debu
	Adding file: bin/Debug/armtestapp
	done
	Registered new type: wxString Registered new type: STL String
	Registered new type: STL Vector
	Debugger name and version: GNU gdb 6.8-debian

Example workspace with Watches turned on.

## Running our new program on the ARM Processor

The next thing we need to do is configure a couple tools so we can communicate and program the OmniFlash ARM processor.

#### **Serial Port Configuration - PuTTY**



Start up PuTTY. (You can use gtkterm also. For the sake of this document, I will only cover PuTTY).

8	PuTTY Configuration	_ <b>- x</b>
Category: Category: - Session - Logging - Terminal - Keyboard - Bell - Features - Reatures - Ronts - Connection - Colours - Fonts - Connection - Data - Proxy - Telnet - Rlogin - Serial	Basic options for your PuTTY se Specify the destination you want to conne Serial line /dev/ttyS0 Connection type:	ect to Speed 57600 Serial
	Close <u>w</u> indow on exit:	Sa <u>v</u> e <u>D</u> elete an exit
About	Open	<u>C</u> ancel

Click the option in the upper right-hand section of the screen. Change the speed to 57600. Give the session a name and then click Save.

8	PuTTY Configuration	X
Category:	Options controllin Set the size of the window	ng PuTTY's window
Logging	Colu <u>m</u> ns	<u>R</u> ows
	80	24
-Bell	Control the scrollback in the	e window
-Features	Lines of scrollback	9999
Window	- (	
Appearance	<u> ⊏ D</u> isplay scrollbar	
Behaviour Translation	□ Scro <u>l</u> lbar on left	

If you want more scroll back lines, load the profile for Serial Port 1 and then change the Window settings for scroll back lines. Then go back to the Session option and save it again.

8	PuTTY Configuration	_ <b> </b>
Category: Session Logging Terminal Keyboard Bell Features Window Appearance Behaviour Translation Selection Colours Fonts Fonts Connection Data Proxy Telnet Rlogin SSH Serial	Basic options for your PuTTY set         Specify the destination you want to connection serial line         /dev/ttyS0         Connection type: <ul> <li><u>Raw</u></li> <li><u>Telnet</u></li> <li><u>Rlogin</u></li> <li><u>SSH</u></li> </ul> Load, save or delete a stored session         Saved Sessions         Serial Port 1         Default Settings         Serial Port 1         Close window on exit:         Always       Never         Only on cle	ession ct to Speed 57600 A Serial Load Save Delete Delete
<u>A</u> bout	<u>Open</u>	<u>C</u> ancel

To open the connection, click the one on the list you want to open, click the Load button, then click the Open button.

## **Connection Verification - PuTTy**



Once we click Open we get the following window. We are ready to turn plug in the OmniFlash and power it up. Connect the serial cable to the OmniFlash and turn on power.



You should get output from the OmniFlash. If you don't get any output, verify you have the OmniFlash plugged in correctly. You may want to try a Windows COM port program like Hyperterminal or Tera Term (<u>http://www.ayera.com/teraterm/</u>) and verify you have everything connected and working. Note, when the Virtual Machine is running, you will NOT be able to use the serial port from Windows.

#### **Serial Port Configuration - CuteCom**

Next we need to configure the program used to send software via X-modem protocol to the OmniFlash. This program is called cutecom. Since a quick launch icon was not created for cutecom, we need to create one.

	Places	System		
Applications	Haces	System		⊕ <u>A</u> dd to Panel
				broperties
				🗑 Delete This Panel
<b>STATE 11 1 1 1</b>			 	

Right-click on the toolbar past the question mark and select Add to Panel...

0	Add to Panel	×				
Find a	n item to add to the panel:					
a a a a a a a a a a a a a a a a a a a	Custom Application Launcher Create a new launcher					
-	Application Launcher Copy a launcher from the applications menu	Ξ				
	Address Book Search Search for a person in your address book					
	Battery Charge Monitor Monitor a laptop's remaining power					
1	Brightness Applet Adjusts Laptop panel brightness					
á	Character Palette Insert characters					
ā	Clipboard Text Encryption Encrypt, decrypt or sign the clipboard (uses PGP type encryption).					
$\odot$	Clock Get the current time and date					
	Connect to Server Connect to a remote computer or shared disk					
0	Help	•				

Choose Custom Application Launcher.

•	Cr	eate Launcher	X
2000	<u> </u>	Application	-
le le	<u>N</u> ame:	Cutecom	
	Comm <u>a</u> nd	cutecom <u>B</u> rowse.	
	Co <u>m</u> ment:	Serial Port Program	
0 <u>H</u> e	lp	Cancel 🖉 🖓 🖓	

The Command is the most important here. Type cutecom. Give it a name and a comment and press OK. Then close the Add to Panel window.

## **Testing connection with CuteCom**

📢 Applications Places System 🥹 📿 🖗		≡ 1
	CuteCom	
Open device Device:	/dev/ttyS0 YParity:	None
Close device Baud rate:	57600 💙 Handshake: 🗆 Sof	ftware 🗌 Hardware
About Data bits:	8 V Open for: 🗹 Rea	ading 🗹 Writing
Quit Stop bits:	1 ✓ Apply settings v	vhen opening
Clear Hex output Log	g to: 🗸	
	III.	
Input:		
Send file Plain	LF line end	Char delay: 1 ms 🗘
		·

Note: Make sure you close PuTTY before you do this as only one program can be connected to the serial port at a time.

We now have a new icon at the top of our screen. Click it and test it to see that it works. Verify that the Device selected in ttyS0 and the baud rate is 57600.

The way this program works is we Open the device (for reading and writing) when we need to communicate with the OmniFlash. We can only send one command at a time on the Input line. Output is shown in the big white square.

		CuteCom			_ O X
Open device	Device:	/dev/ttyS0	~	Parity:	None 🗸
Cl <u>o</u> se device	Baud rate:	57600	$ $ $\vee$	Handshake: 🔲 Softwa	are 🔲 Hardware
About	Data bits:	8	<b> </b> ~	Open for: 🗹 Readin	ng 🗹 Writing
Quit	Stop bits:	1	<b> </b> ~	Apply settings whe	en opening
m					<u>^</u>
drwxr-xr-x       2         drwxr-xr-x       3         drwxr-xr-x       4         drwxr-xr-x       2         drwxr-xr-x       2         drwxrwxrwt       2         drwxr-xr-x       5         drwxr-xr-x       5         ~ #	root root root root	1024 Sep 22 2048 Jan 1 1024 Nov 16 1024 Dec 18 2048 Dec 18 1024 May 31 0 Jan 1 1024 Jan 18 1024 Dec 18 1024 Dec 18 1024 Dec 18	2005 1970 2005 2004 2004 2005 1970 2006 2004 2004 2004	\ox1b[1;34mbin\0x1b \ox1b[1;34mdev\0x1b \0x1b[1;34mdev\0x1b \0x1b[1;34metc\0x1b \0x1b[1;34mhome\0x1b \0x1b[1;34mlib\0x1b \0x1b[1;34mproc\0x1b \0x1b[1;34mproc\0x1b \0x1b[1;34mtmp\0x1b \0x1b[1;34mtmp\0x1b \0x1b[1;34mtmr\0x1b \0x1b[1;34mtmr\0x1b \0x1b[1;34mtmr\0x1b \0x1b[1;34mtmr\0x1b \0x1b[1;34mtmr\0x1b	b     [Om       b     [Om
ls Is -al Input Is -al					
Send file Plain	<b>\</b>		(	LF line end 🗸 Cha	r delay: 1 ms 🗘

On the input line, we can type commands. Pressing Enter will send them. Once we send a command, we can double-click it from the lower window list.

#### Sending a program via CuteCom

	drwxr-xr-x 5 root	roo
(	drwxr-xr-x 5 roat	roo
	CCCCCC	
C	<u>C</u> lear <u>H</u> ex output	
	ls	
	ls -al	
(	rx armtestapp	
	Input:	
(	Send file XModem 🗸	)

The first thing we do is type "rx armtesetapp" or whatever the name of the program we want to receive on the ARM processor. We then press ENTER. This sends the receive X-Modem command to the ARM processor. The ARM processor starts querying for the file to be received. You will see at the top of the screen shot that the ARM received our command and the CCCC lets us know it is retrying to receive the file. Next we change the file type to XModem and click the Send file... button.

Note: We can only save files to **/mnt/FlashMemory** if we want them to persist when we reboot. For this test, we won't be writing to this location. We will send it to the root file system which will be erased when we reboot the OmniFlash.

D	Open 🛛
Look in:	🗎 /home/" 💓 /code/armtestapp/bin/armRelease 🗸 🗸 🖨 📄 🔒 📰 🔳
Compute	rr
File <u>n</u> ame:	armtestapp Open
Files of type:	All Files (*)

Navigate to where the program we built earlier resides. Look in the armRelease folder for the application.

Click Open to send the file.

	diwal-Al-A	3 1000	1000	1024	Dec 10	2004	(OXID[1, SAUGSI (OXID[OU	
	drwxr-xr-x	5 root	root	1024	1 Dec 18	2004	\0x1b[1;34mvar\0x1b[0m	. 1
ſ	~ # rx armtes	stapp						٦.
	CCCCCCCCC\02	15\0x15\0x	15\0x15\0x	15\0x15\0	x15\0x15	\0x15	\0x18\0x18\0x18\0x18\0x18\0x0	8
	\0x08\0x08\0x	08\0x08rx:						
	rx: receive failed:							
	too many errors; giving up							
l	~ #							-
1			_					_
	Clear 🗌	Hex output	L	.og to:	✓ /hom	e/natha	an/cutecom.log	
	_			-		-		_

If the ARM processor times out, you will have to send it again.

Open device	Device:	/dev/ttyS0	Y Parity:		
Cl <u>o</u> se device	Baud rate:	57600	Handshake: 🗆 So		
About	Data bits:	8	✓ Open for: ✓ Re		
Quit	Stop bits:	1	✓ ✓ Apply settings		
<pre>drwxr-xr-x 2 drwxrwxrwt 2 drwxr-xr-x 5 drwxr-xr-x 5 drwxr-xr-x 5 ~ # rx armtestapp CCCCCCCCCC(0x15) \0x08\0x08\0x08\0x08\ rx: receive fails too many error. ~ # rx armtestapp C~ # rx armtestapp C</pre>	root root root root root root root root p 0x15\0x15\0x15 0x08rx: ed: s; giving up p pp	1024 Jan 18 cutecom ending file via xmod 2% Cand	2006 \0x1b[1;34msbin 0x1b[1;34mtmp\0 0x1b[1;34mtmp\0 0x1b[1;34musr\0 0x1b[1;34mvar\0 x1b[1;34mvar\0 x18\0x18\0x18\0x18\0		
<u>Clear</u> <u>Hex output</u> Log to: ∨ /home/nathan/cutecom.log					
		m			
ls ls -al rx armtestapp <u>I</u> nput: rx armtestapp	)				
Send file XMod	em 🗸		LF line end		

Here, I double-clicked on the "rx armtestapp" text which sent it to the ARM processor. Then I clicked the "Send file..." button and selected my test application. You can see that the file is being transferred to the ARM processor.

D		
	<u>O</u> pen device	Device:
	Cl <u>o</u> se device	Baud rate:
	<u>A</u> bout	Data bits:
	<u>Q</u> uit	Stop bits:

Click the "Close device button after you have sent the file.

#### Verifying the program was received by the OmniFlash

dev/ttys0 - Pully	/dev/ttyS0 - PuTTY				
<pre>" # ls armtestapp dev home mnt sbin usr bin etc lib proc tmp var " # </pre>	home mnt lib proc	sbin usr tmp var			

Now launch PuTTY to connect to the OmniFlash again. Press enter to get a prompt back. Type "Is" and you should see the file we just transmitted.

#### Launching the program



Before we can run the application, we need to set "execute" permissions on the file.

Type "**chmod +x armtestapp**" to make it an executable

Now try running it and see what happens.



Type "**./armtestapp**" to run it. The "./" tells the operating system to look in the current directory (since it isn't in the path).

Notice we ran the ARM specific code and not the PC specific version.

There you have it. A fully graphic environment to code and test your application for your OmniFlash. And we didn't have to write a single makefile to compile for the ARM processor.

This is a bit of a juggling act in that you have to switch back and forth between two applications remembering to close PuTTY when you use CuteCom and remembering to close the connection on CuteCom before you can use PuTTY. But at least you don't have to type all of this from the command line.

#### **Final Notes**

#### Note about writing to /mnt/FlashMemory.

If you write your programs to /mnt/FlashMemory, be sure to type "sync" to write it to flash memory after you chmod the program or anytime you write to the Flash memory system. If you don't and you lose power or reboot your device, you may corrupt the Flash Memory and you will have to repair it.