# Create C/C++ project for ARM with Eclipse Autotools

www.letrungthang.blogspot.com

### 1. Environment setup

Assuming that you installed and setup GCC for ARM. Else, reference here.

Be sure that you installed automake, autoconf, autoreconf. Else, you need install it.

To install Automake: sudo apt-get install automake

This also will install Autoreconf and Autoconf for you.

To check version of automake: automake --version.

```
thangle@ubuntu:~$ automake --version
automake (GNU automake) 1.14.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv2+: GNU GPL version 2 or later <http://gnu.org/lic/

This is free software: you are free to change and redistribute
There is NO WARRANTY, to the extent permitted by law.

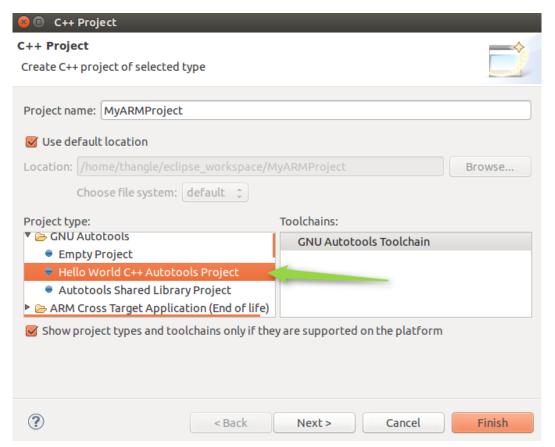
Written by Tom Tromey <tromey@redhat.com>
and Alexandre Duret-Lutz <adl@gnu.org>.
```

### 2. Create an C++ Autotools project

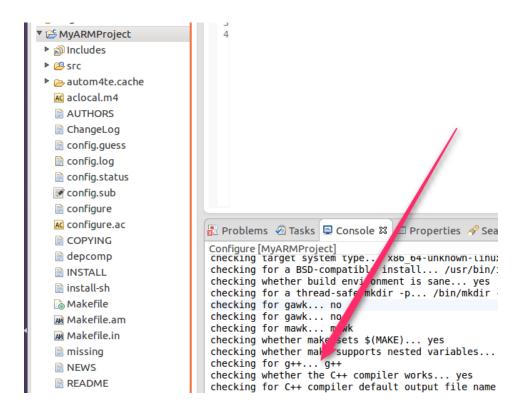
Assuming that your GCC for ARM is installed at:

/Home/CodeSourcery/Sourcery\_CodeBench\_Lite\_for\_ARM\_GNU\_Linux/ Also assuming you are creating a mixed C/C++ project for AT91SAM9260 SoC (Core arm926ej-s).

- In Eclipse, select **File -> New -> C++ Project**. Select *GNU Autotools*.



Click Finish to complete project creation. You will have a new created project as below.



- Now, Autotools still takes g++ (x86) as default compiler. So, you need tell Autotools to take GCC for ARM when it builds your project. Go to C/C++ Build Environment and configure variables as below.

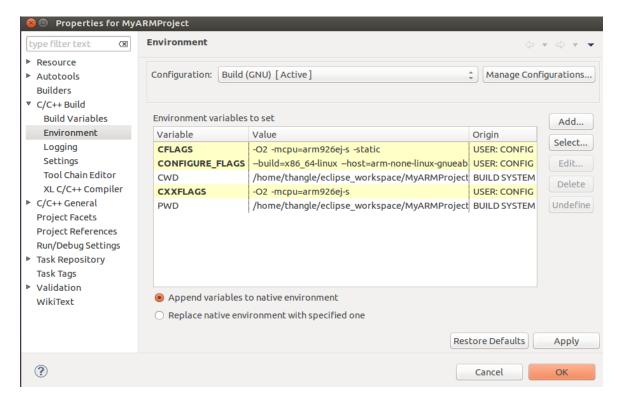
CONFIGURE\_FLAGS = --build=x86\_64-linux --host=arm-none-linux-gnueabi --target=arm-none-linux-gnueabi --CROSS\_COMPILE=arm-none-linux-gnueabi --ARCH=arm

```
CXXFLAGS = -02 -mcpu=arm926ej-s
CFLAGS = -02 -mcpu=arm926ej-s -static
```

**Note**: You need to change the option –mcpu to respective CPU in your target.

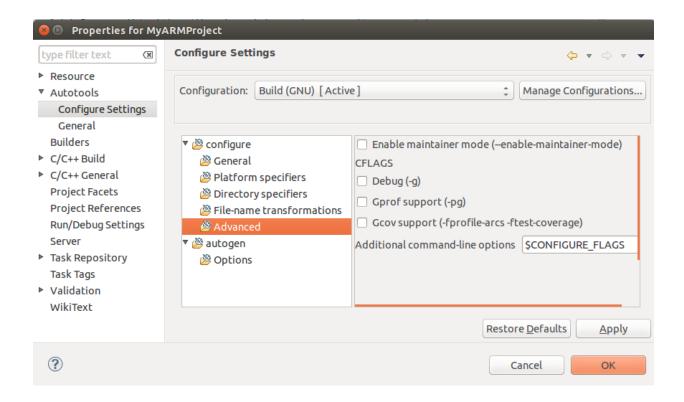
For example: -mcpu=cortex-a15 for ARM Cortex A15 core. More info reference in GCC page <a href="here">here</a>. Below is list of CPUs supported by Sourcery CodeBench Lite 2013 (arm-2013.05-24-arm-none-linux-gnueabi).

arm1020e arm1020t arm1022e arm1026ej-s arm10e arm10tdmi arm1136j-s arm1136jf-s arm1156t2-s arm1156t2f-s arm1176jzf-s arm2 arm250 arm3 arm6 arm60 arm600 arm610 arm620 arm7 arm70 arm700 arm700i arm710 arm7100 arm710c arm710t arm720 arm720t arm740t arm7500 arm7500fe arm7d arm7dm arm7dmi arm7dmi arm7tdmi arm7tdmi-s arm8 arm810 arm9 arm920 arm920t arm922t arm926ej-s arm940t arm946e-s arm966e-s arm968e-s arm9e arm9tdmi cortex-a15 cortex-a5 cortex-a7 cortex-a8 cortex-a9 cortex-m0 cortex-m1 cortex-m3 cortex-m4 cortex-r4 cortex-r4f cortex-r5 ep9312 fa526 fa606te fa626 fa626te fa726te fmp626 generic-armv7-a iwmmxt iwmmxt2 mpcore mpcorenovfp native strongarm strongarm110 strongarm1100 strongarm1110 xscale

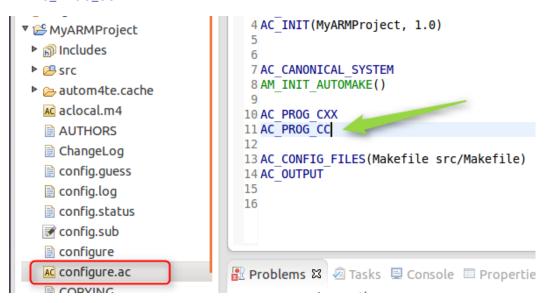


- Next is to setup configure for the project.

In Eclipse, go to project **Properties -> Autotools -> Configure Settings -> Configure -> Advanced** Put **\$CONFIGURE\_FLAGS** to *Additional command-line options* box.



 Because the project mixed both C and C++ code, so to add 2 instructions below to the file configure.ac AC\_PROG\_CXX AC\_PROG\_CC



So far, you still don't have any C source file in your project, so just add it. For simple, I add a C source file is **MyFunction.c** with only a simple function as below.

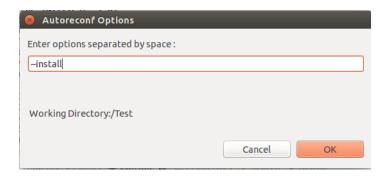
```
9 #include <stdio.h>
▼ S MyARMProject
                             10
 ▶ ⋒ Includes
                             11⊖ int MyFunction()
 ▼ 🕮 STC
                             12 {
                                  printf("MyFunction in C\n");
                             13
  ▶ MyARMProject.cpp
                             14
                                  return 0;
  ▶ 🖟 MyFunction.c
                             15
                             16
    Makefile
    Makefile.am
   Makefile.in
```

You need inform Autotools about this new C source file by adding it to file makefile.am



## 3. Build project

- In Eclipse, right click and run Invoke Autotools -> Invoke Autoreconf and type --install in Autoreconf box. This is requested only to do one time when you create your project. Automake will install necessary things for your project.



- Next, In Eclipse right click and run **Reconfigure Project**. This only need do once when you create new project and you change something in makefile.am, configure.ac.

If build successful, you could see messages as below

```
▶ S Algorithms
                              🚼 Problems 🔑 Tasks 星 Console 🛭 🗏 Properties 🔗 Search 🕸 Debug
▼ S MyARMProject
                              Configure [MyARMProject]
 ▶ ⋒ Includes
                              checking whether build environment is sane... yes
 ▶ # SFC
                              checking for arm-none-linux-gnueabi-strip... arm-none-linux-gnueabi-strip
                              checking for a thread-safe mkdir -p... /bin/mkdir -p
 ▶ autom4te.cache
                              checking for gawk... no
   aclocal.m4
                              checking for gawk... no
                              checking for mawk... mawk
   AUTHORS
                              checking whether make sets $(MAKE)... yes
   ChangeLog
                              checking whether make supports nested variables... yes
   compile
                              checking for arm-none-linux-gnueabi-g++... arm-none-linux-gnueabi-g++
                              checking whether the C++ compiler works... yes
   config.guess
                              checking for C++ compiler default output file name... a.out
   config.log
                              checking for suffix of executables...
   config.status
                              checking whether we are cross compiling... yes
                              checking for suffix of object files... o
   config.sub
                              checking whether we are using the GNU C++ compiler... yes
   configure
                              checking whether arm-none-linux-gnueabi-g++ accepts -g... yes
   ac configure.ac
                              checking for style of include used by make... GNU
                              checking dependency style of arm-none-linux-gnueabi-g++... gcc3
   COPYING
                              checking for arm-none-linux-gnueabi-gcc... arm-none-linux-gnueabi-gcc
   depcomp
                              checking whether we are using the GNU C compiler... yes
   INSTALL
                              checking whether arm-none-linux-gnueabi-gcc accepts -g... yes
                              checking for arm-none-linux-gnueabi-gcc option to accept ISO C89... none needed
   install-sh
                              checking whether arm-none-linux-gnueabi-gcc understands -c and -o together... yes
   Makefile
                              checking dependency style of arm-none-linux-gnueabi-gcc... gcc3
   Makefile.am
                              checking that generated files are newer than configure... done
                              configure: creating ./config.status
   Makefile.in
                              config.status: creating Makefile
   missing
                              config.status: creating src/Makefile
                              config.status: executing depfiles commands

■ NEWS

                              [Operation successful]
   README
```

Last, in Eclipse right click on top of project and select Build Project.
 If build successful, you will see messages below.

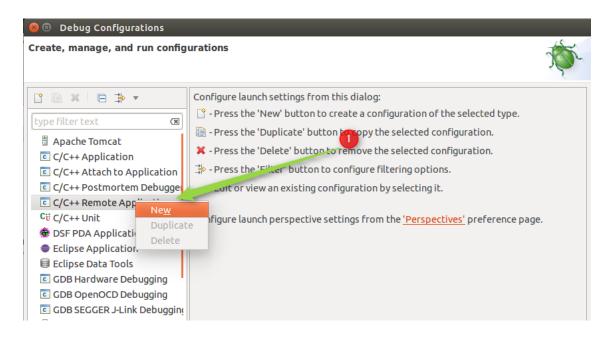
```
🖳 Problems 🙆 Tasks 📮 Console 🏻 🗏 Properties 🔗 Search 🕸 Debug
                                                                                                                     CDT Global Build Console
23:35:01 **** Build of configuration Build (GNU) for project MyARMProject ****
make all
Making all in src
make[1]: Entering directory `/home/thangle/eclipse workspace/MvARMC.oject/src'
arm-none-linux-gnueabi-g++ -DPACKAGE NAME=\"MyARMProject)
                                                                     ■PACKAGE TARNAME=\"myarmproject\" -DPACKAGE VER
mv -f .deps/MyARMProject.Tpo .deps/MyARMProject.Po
arm-none-linux-gnueabi-gcc -DPACKAGE_NAME=\"MyARMProject\" -DPACKAGE_TARNAME=\"myarmproject\" -DPACKAGE_VER
mv -f .deps/MyFunction.Tpo .deps/MyFunction.Po-
arm-none-linux-gnueabi-g++ -02 -mcpu=arm926ej-s a out MyARMProject.o Mmake[1]: Leaving directory `/home/thangle/eclipse_workspace/MyARMProject/src'make[1]: Entering directory `/home/thangle/eclipse_workspace/MyARMProject'
                                                                 out MyARMProject.o MyFunction.o
make[1]: Nothing to be done for `all-am'.
make[1]: Leaving directory `/home/thangle/eclipse workspace/MyARMProject'
23:35:02 Build Finished (took 969ms)
```

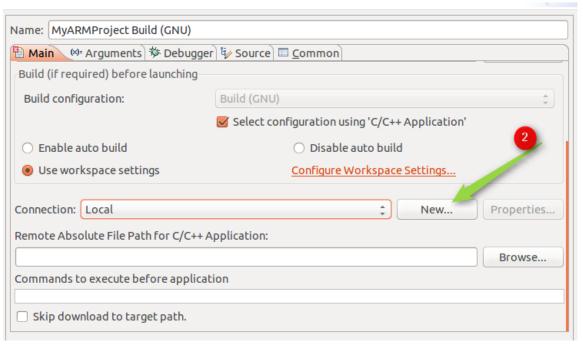
A binary (a.out) is generated. It is executable file that can run on your target device.

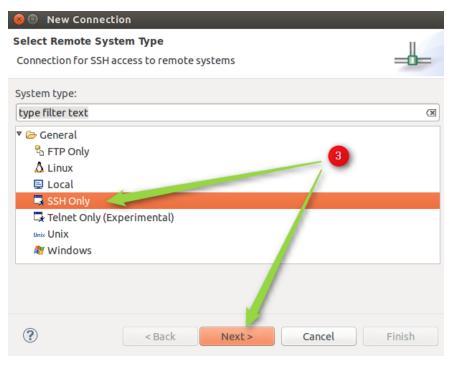
```
4 AC INIT(MyARMProject, 1.0)
🗸 😂 MyARMProject
 ▼ ₩ Binaries
  * a.out - [arm/le]
                                 7 AC CANONICAL SYSTEM
                                 &AM INIT AUTOMAKE()
 ▶ ⋒ Includes
 ▶ 2 src
                                10 AC PROG CXX
                                11 AC PROG CC
 autom4te.cache
                                12
  AC aclocal.m4
                                13 AC CONFIG FILES(Makefile src/M
  AUTHORS
                                14 AC OUTPUT
                                15
  ChangeLog
                                16
  compile
```

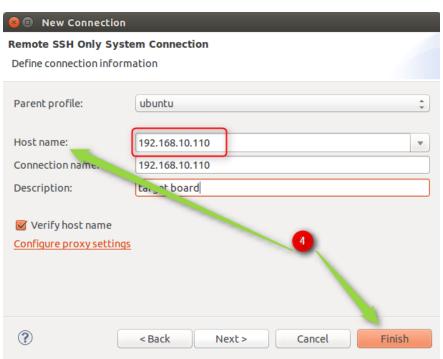
## 4. Run and Debug your program

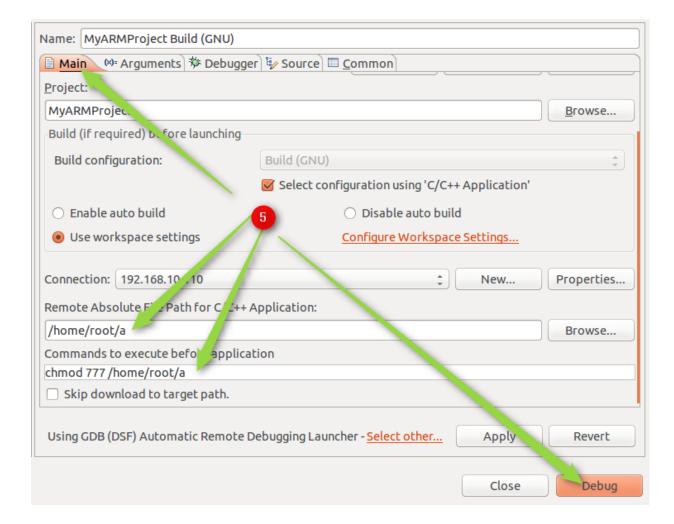
You can copy generated binary to target and run it manually from your target or you can run directly it from Eclipse by configuring as below if you are able to connect to your target via Ethernet. You can debug only if your target installed gdb-server. Below is steps to configure Eclipse debugger











### 5. References

- Example source code in Git hub.
- Autotools manual

