Setting up development environment

Notes: executable files can be installed graphically or by command lines

1. GNU make

$ sudo apt install build-essential checkinstall

2. GNU toolchain for ARM Cortex-M

Download and install the GNU ARM Toolchain package for linux-64

```
$ sudo apt install unzip
$ tar xf gcc-arm-none-eabi-7-2017-q4-major-linux.tar.bz2
```

// may need gcc_multilib

```
$ sudo apt install gcc-multilib
```

// add the location to the path temporarily when executing locally in a terminal:

```
$ export PATH=$PATH: $(HOME)/.../.../gcc-arm-none-eabi-7-2017-q4-major/bin
$ echo $PATH
```

3. SEGGER J_Link

Download and install the Segger JLink package for linux-DEB-64

```
$ sudo dpkg -i JLink_Linux_V632g_x86_64.deb
```

//check exe file "jlink*" under usr/bin

4. nRF5_SDK

Download and unzip the nRF52-SDK zip package

//To do make, need to source GNU ARM

nRF52_SDK >>components >> toolchain >> gcc >> Makefile.posix

```
GNU_INSTALL_ROOT := $(HOME)/.../.../gcc-arm-none-eabi-7-2017-q4-major/bin/
yGNU_VERSION := 7.2.1
GNU_PREFIX := arm-none-eabi
```

5. nRF5x-Command-Line-Tools

Download the nRF5x Tools Linux64 (see p.10)

```
$ gedit ~/.bashrc
    export PATH=$PATH:~/.../.../nRF5x-Command-Line-Tools/nrfjprog
    export PATH=$PATH:~/.../.../nRF5x-Command-Line-Tools/mergehex
```

6. Reset MCU interface

JLINK interface is set as default interface for the dev kit and chosen to be used in this project. Other MCU interface can be used, for example DAPLINK interface.

JLINK interface: Download .bin file for JLINK interface software from: http://www.nordicsemi.com/eng/Products/nRF52840-DK#Downloads

DAPLINK interface: Download .bin file for DAPLINK interface software from os.mbed.com (https://os.mbed.com/platforms/Nordic-nRF52840-DK/)

- Press boot/reset button while turning power switch from OFF to ON -> BOOTLOADER mode
- Copy the downloaded .bin file into devboard and wait until LED5 fast flashing
- Press boot/reset button while turning power switch from OFF to ON -> MCU interface (JLINK/DAPLINK etc.) is reset.

7. Run demo with GCC and command tools (JLink interface)

Change directory to ~/.../.../sdk/examples/peripheral/blinky/pca10056/blank/armgcc/
//do make, _build folder is created
$ make

/*1. run with nrfjprog*/
```
//erase target
$ nrfjprog --family nRF52 -e
//load FW image to target
$ nrfjprog --family nRF52 --program _build/nrf52840_xxaa.hex
//reset and run
$ nrfjprog --family nRF52 -r
```

/*2. run with JLinkExe*/
```
/* Open Jlink Commander from terminal in _build directory */
JLinkExe -device <nRF51/nRF52>
> erase // Optional: erase target if not already blank
> loadfile <name>.hex // loads FW
> r // Reset and halt
> g // Run
> q //  Exit
```

/*3. directly copy example .hex file into target via USB*/

8. Program target with os.mbed.com/compiler

- Target device: nRF52-DK
- Import demo project *nRF52840-Preview-DK-blinky*
- Use mbed.h library for high level programming
- Code and compile to create application .hex file
- Copy the .hex file to the target and wait until the JLINK/DAPLINK interface disappears and reappears
- Press boot/reset button to start running the application

Good reference -> import BLE project: [mbed-os-example-client-BLE-NRF52_DK]

9. Setting up Eclipse IDE

******Download and install Eclipse******

- Download and unzip eclipse package (newest version of Eclipse Oxygen linux 64 bit): http://www.eclipse.org/downloads/
- Double click on "eclipse-inst" exe file >> Eclipse IDE for C/C++ Developers >> choose directory for installation
- Double click on "eclipse" exe file >> choose directory for workspace
- Source exe file:
  $ gedit ~/.bashrc
  export PATH=$PATH:~/.../.../eclipse

Install GNU MCU eclipse plug in

Open eclipse in terminal or by double click on the exe file >> Help >> Eclipse Marketplace >> Find and install GNU MCU Eclipse
Required packages are:
GNU ARM C/C++ Cross Compiler
GNU ARM C/C++ Packs
GNU ARM C/C++ J-link Debugging
All packages are chosen by default.

Install device family pack for the nRF5x serie

Window >> Perspective >> Open Perspective >> Other >> Packs >> Open.
Click the refresh button to fetch all packs from the repositories.
Select Nordic Semiconductor in the list of vendors, and install the latest version of Device family pack. (Skip if all invalid packages!)

Create a project

Common Settings:

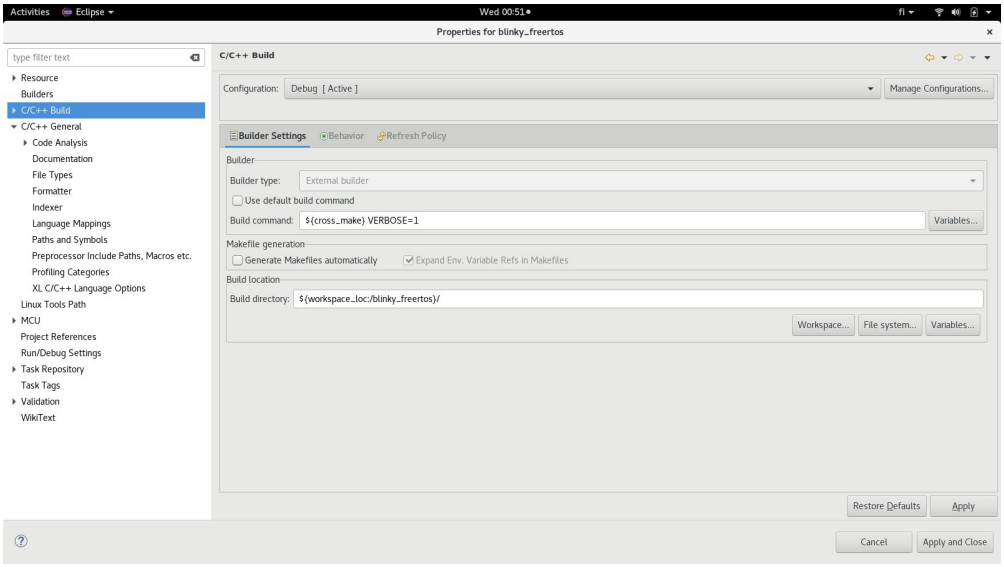
- Window >> Preference >> MCU
  - >> Global ARM Toolchains Paths >> Default toolchain: *GNU Tools for ARM Embedded Processors* >> Build tools folder: *~/.../.../gcc-arm-none-eabi-7-2017-q4-major/bin*
  - >> Global Build Tools Path >> Build tools folder: ~/nRF5x-Command-Line-Tools/nrfjprog
  - >> Global SEGGER J-Link Path >> Executable: JLinkGDBServerCLExe >> Folder: /opt/SEGGER/JLink (default settings)

Launch example project blinky for nrf52840:

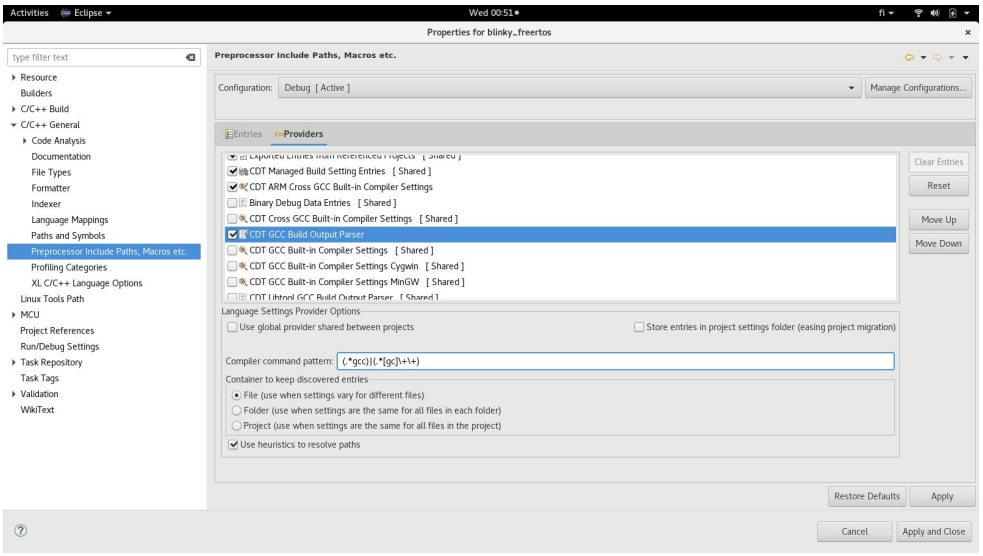
File >> New >> Project >> C/C++ >> C Project
Project name: blinky
Project type: Empty Project
Toolchains: ARM cross GCC
Next all with other settings as default until the project “blinky” folder appears in Project Explorer.

Project Settings:

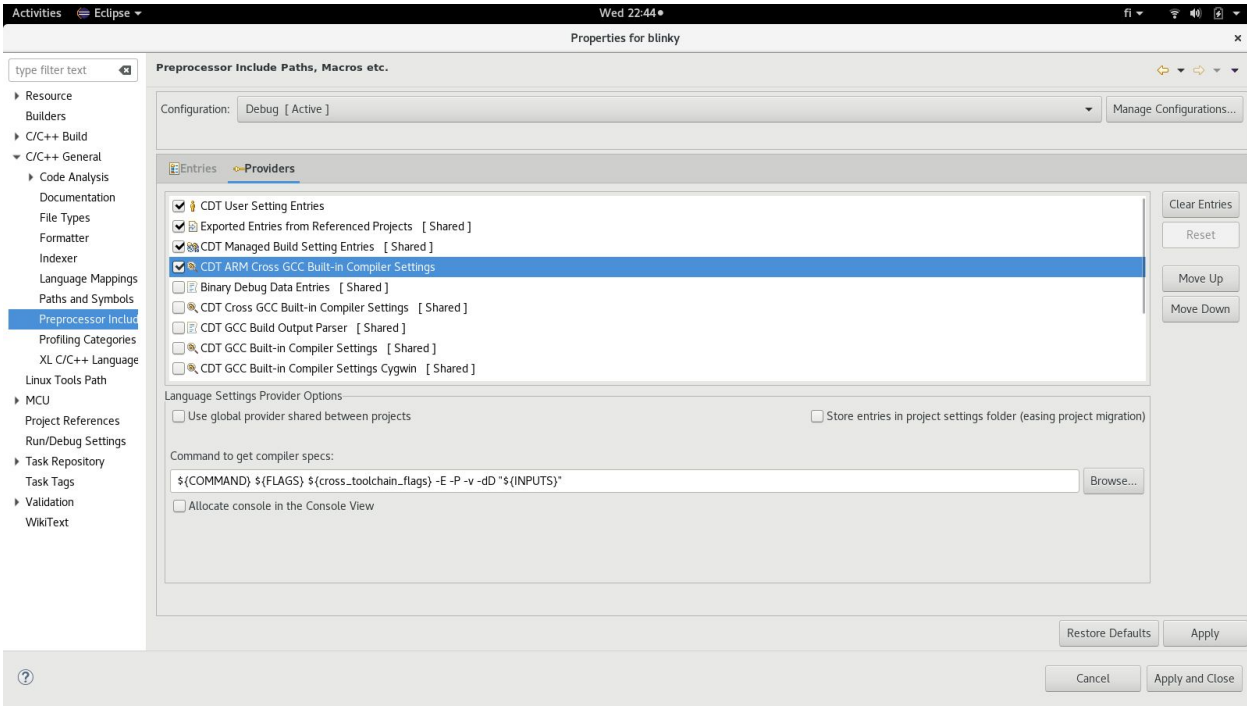
- Double click on the project folder (blinky) >> Properties
  - >> C/C++ Build >> deselect Use default build command >> Build command: ${cross_make} VERBOSE=1 >> deselect Generate Makefile automatically >> Build directory: ${workspace_loc:/blinky}/



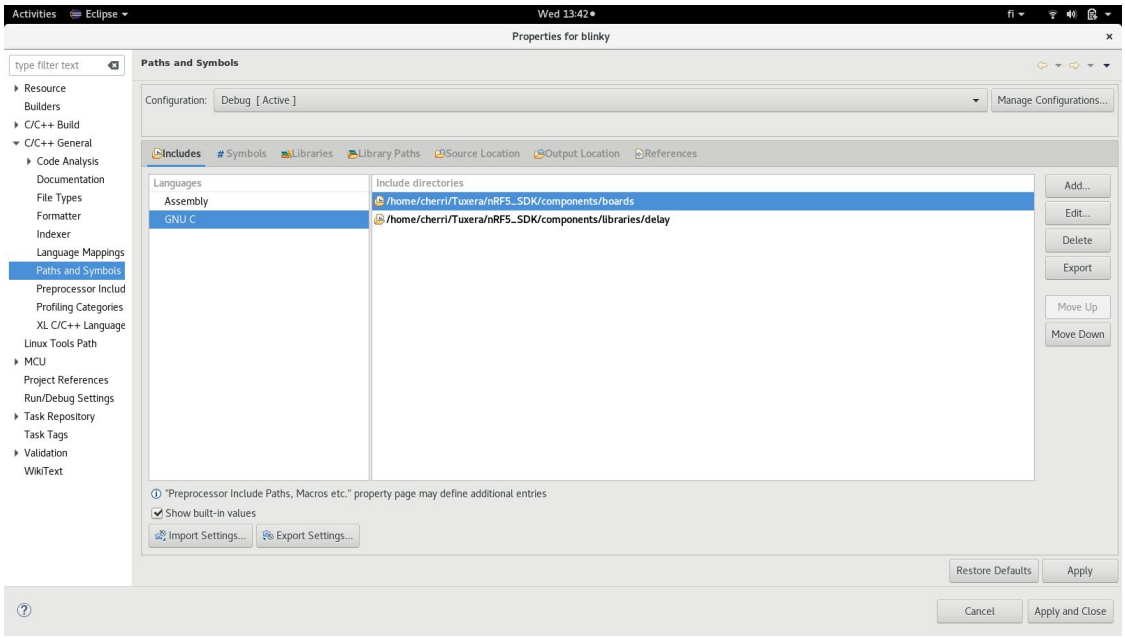
  - >> C/C++ General >> Preprocessor Include Paths,Macros etc.
    - >> CTD GCC Build Output Parser >> Compiler command pattern: (.*gcc)|(.*[gc]\+\+)



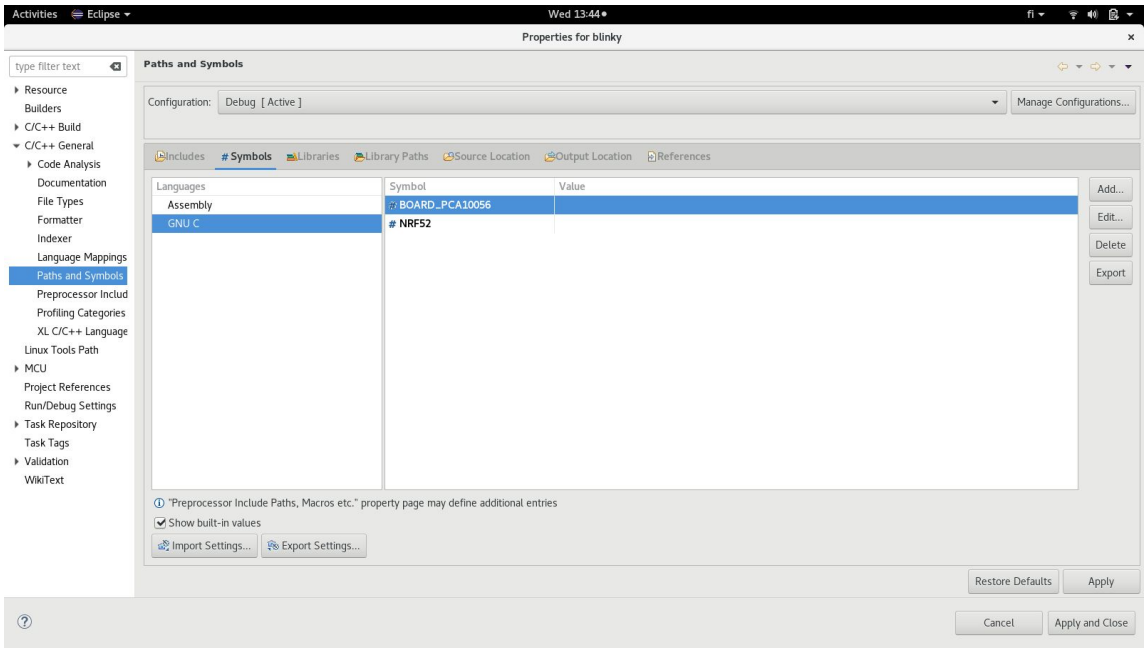
    - >> CDT ARM Cross GCC Built-in Compiler Settings >> Command to get compiler specs: (keep as default)



- >> C/C++ General >> Paths and Symbols >> Includes >> Add (Add to all configurations)

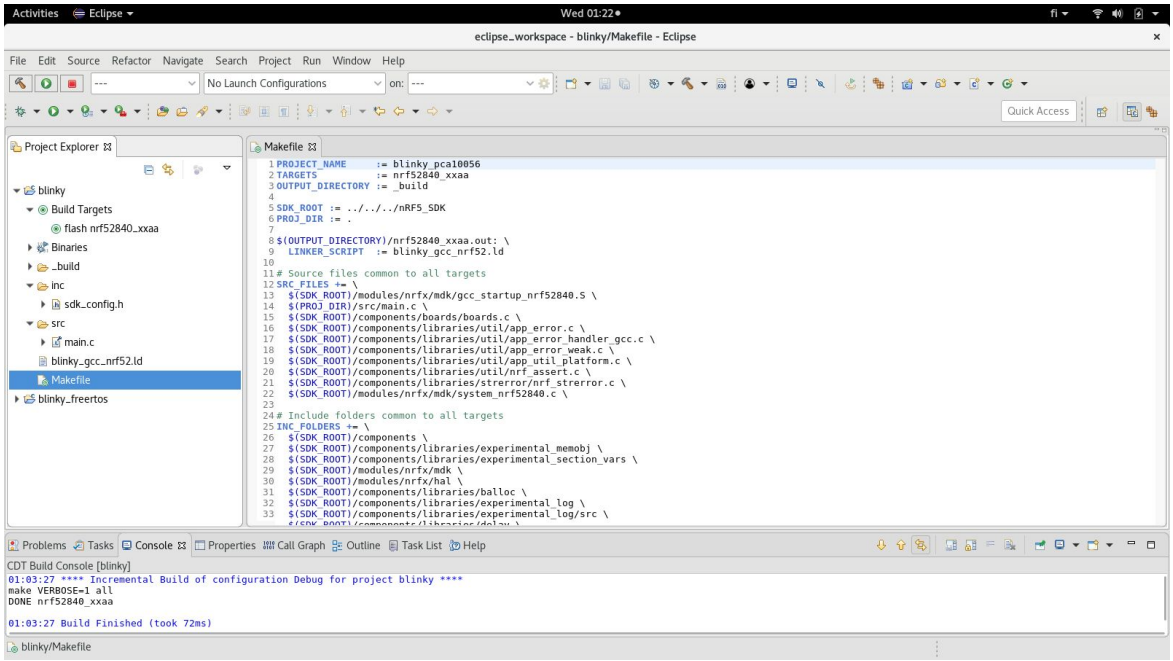


- >> C/C++ General >> Paths and Symbols >> Symbols >> Add (Add to all configurations)



- Import source files from nRF5_SDK/examples/peripherical/blinky/
  - main.c -> src/main.c
  - pca10040/config/sdk_config.h -> inc/sdk_config.h
  - pca10040/blank/armgcc/Makefile -> Makefile
  - pca10040/blank/armgcc/blinky_gcc_nrf52.ld -> blinky_gcc_nrf52.ld

- Edit Makefile:
  - SDK_ROOT := ../../../sdk (exact location of SDK)
  - PROJ_DIR := . (location where the project is built)
  - $(PROJ_DIR)/main.c \ -> $(PROJ_DIR)/src/main.c \
  - $(PROJ_DIR)/ \ -> $(PROJ_DIR)/inc \
  - Remove ../config \
  - For CFLAGS, OPT = -O0 -g3



Build project:
Right click on the project folder >> Build Project

Build target and flash download:
Double click on the project folder >> Build Targets >> Create >> Target name: flash nrf52840_xxaa
Right click on the project folder >> Build Targets >> Create >> Target name: flash nrf52840_xxaa
Right click on the target >> Build Target

10. Debugging
RTT debugging at terminals:
In one terminal:
$ JLinkExe -if swd -device nrf52 -AutoConnect 1 -speed 4000
In another terminal:
$ JLinkRTTClient
Links:
https://devzone.nordicsemi.com/tutorials/b/getting-started/posts/development-with-gcc-and-eclipse
https://gnu-mcu-eclipse.github.io/debug/jlink/
https://devzone.nordicsemi.com/f/nordic-q-a/14512/how-to-use-rtt-viewer-or-similar-on-gnu-linux