\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Installation Guide for the BMD-300 eval board with nRF52 using OpenOCD and nRF5\_SDK on Ubuntu 16.04

 Written by Zoltan Bardos - Sep 21, 2017

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

INTRO:

This guide assumes an EARLY BEGINNER to terminal commands, Linux, Git, and to the software packages used, such as myself. I am writing up this guide for myself, and any interns that come after me, and anyone else who stumbles upon it online. This guide is in response to an issue detailed in a forum post made by me here:

https://devzone.nordicsemi.com/question/168973/nrf52-nrfjprog-error-there-is-no-debugger-connected-to-the-pc/

If you havent already, install a Linux distribution. I used Ubuntu 16.04.

I am using the dev board to program and test a Bluetooth type board develpoed by my team, which uses an nRF52 chip.

------------------\_\_\_\_\_\_------ (1) nRF52 SDK ------------------------------------

The first step is to download nRF52 SDK tools: nRF5\_SDK\_11.0.0.89a8197.zip from:

https://developer.nordicsemi.com/nRF5\_SDK/

To unzip a .zip file in linux terminal, navigate to the folder containing the .zip using 'cd', for example:

 'cd /home/Downloads'

Run:

 'unzip [FILE NAME.zip] -d [DESTINATION\_FOLDER]'

If it yells at you about "unknown command unzip", run:

 'sudo apt-get install unzip'

------------------------------- (2) ARM Embedded tools ---------------------------

Next we need to install ARM Embedded Toolchain. The package can be found at:

https://launchpad.net/~team-gcc-arm-embedded/+archive/ubuntu/ppa

Use the provided install comands:

 'sudo add-apt-repository ppa:team-gcc-arm-embedded/ppa'

 'sudo apt-get update'

EDIT: Apparently, this isnt good enough, so I had to do:

 'sudo apt-get install gcc-arm-none-eabi'

-------------------------------- (3) OpenOCD ----------------------------------

Clone and patch OpenOCD. A forum post describing the patch can be found here:

https://devzone.nordicsemi.com/question/78890/programming-nrf52-with-openocd/

Navigate to the folder you want to save this repository in, and run:

(If it yells at you about "unknown command Git", 'sudo apt-get install git')

 'git clone git://git.code.sf.net/p/openocd/code openocd-code'

Next cd into the repo with...

 'cd openocd-code'

Next patch it by pulling the following patch:

 'git pull http://openocd.zylin.com/openocd refs/changes/15/3215/2'

There will probably be an 'auto merge conflict'. To manually fix the conflict, run:

 'gedit src/flash/nor/Makefile.am'

 'gedit src/flash/nor/drivers.c'

OR just navigate to the indicated files in your file browser and open them. conflice in Git will modify these files to contain BOTH versions of the code, and we will decide what to keep and delete.

\*\*\* In Makefile.am \*\*\*

Find '<<<<<<< HEAD' in the code. This is NOT part of the code. this was added by Git to indicate wehre the conflict exists. Everything between the first '<<<<<<< HEAD' and '=======' is what existed in your current repository branch. Everything between '=======' and '>>>>>>> 68611a6-blah-long-hash-4f22bb' is what is causing conflict in the branch you just tried to pull and auto-merge. We need will edit this file by hand and then commit the changes to our local branch.

First find 'nrf52.c \' in the pulled code. Append it into the code we already had in the '>>>HEAD part'. Make sure to add the '%D%/' similar to the others in the list. Then delete everthing else in the pulled branch part. We only needed the 'nrf52.c'. Also delete the '>>>>HEAD' '=====' '>>>>6B9s38' marker lines, and the extra line they made.

\*\*\* In drivers.c \*\*\*

Similar process. In this file, you can see that there is no existing code in the first '>>>HEAD' section, so just delete the markings there. As for the code being added by the pulled branch, leave 'nrf52\_flash' alone, but you do need to delete the duplicate structs 'numicro\_flash' and 'nrf51\_flash' or it will throw an error on compiling about duplicate references. They already exist in the list. There is another conflict further down the file. Do the same. Remove the duplicates, leave the 'nrf52\_flash' part.

Save the files, close them, and run in terminal:

 'git add -A'

...to add ALL files Git recognizes as changed to the commit staging and...

 'git commit -m"comment about the commit"'

...to commit the changes and comment.

Next we get to build/install OpenOCD! Run './bootstrap' ...But oops, it will demand we have libtool. You can search for it using:

 'apt-cache search libtool'

then install it:

 'sudo apt-get install libtool'

It then demands 'aclocal'. I fixed this by also installing 'autotools-dev' and 'automake'

Then

 './bootstrap'

...should work!

Before we can run the next huge command, it will yell at us to install:

libusb-1.0-0-dev

libhidapi-dev

libftdi-dev

After those are installed, run this command, but MAKE SURE the path is changed to match your path to your openocd clone. My path, for example, was was '/home/ztanb/Projects/Flare/openocd-code':

(FYI you can copy & paste into terminal with ctl-shift-c and ctl-shift-v)

 !!! Make sure your path is correct after 'prefix=' !!!

Once you changed this command appropriatly, run it! (most of these are probably unececsary, but i enabled them all just to make sure.)

'

./configure \

 --prefix=/home/ztanb/Projects/Flare/openocd-code/openocd-git\_install \

 --enable-aice \

 --enable-amtjtagaccel \

 --enable-armjtagew \

 --enable-cmsis-dap \

 --enable-dummy \

 --enable-ftdi \

 --enable-gw16012 \

 --enable-jlink \

 --enable-jtag\_vpi \

 --enable-opendous \

 --enable-openjtag\_ftdi \

 --enable-osbdm \

 --enable-legacy-ft2232\_libftdi \

 --enable-parport \

 --disable-parport-ppdev \

 --enable-parport-giveio \

 --enable-presto\_libftdi \

 --enable-remote-bitbang \

 --enable-rlink \

 --enable-stlink \

 --enable-ti-icdi \

 --enable-ulink \

 --enable-usb-blaster-2 \

 --enable-usb\_blaster\_libftdi \

 --enable-usbprog \

 --enable-vsllink

'

Awesome! now run:

 'make' (this will run an while and spit out a ton of gibberish)

...and...

 'make install'

-------------------------------- (4) Firmware -----------------------------------

Now you just need to have handy or clone your relevent firmware!

Navigate to the project you are trying to flash. For me, I was running a provided example located at '.../nRF5\_SDK\_11.0.0/examples/ble\_peripheral/ble\_app\_beacon/pca10040/s132/armgcc'

Run:

 'make'

------------------------------ (5) Fixing the binary files ----------------------

Still in the example directory, I needed to merge some binary files or something. The following is a record of the exact commands executed by another more experienced engineer. I dont fully understand all the commands:

Install:

 'sudo apt-get install srecord'

 'make'

 'make merge'

(plugs in the board)

 'make flash\_combined'

(for some reason openocd wasnt installed)

(navigates to openocd, runs...)

 './bootstrap'

 './configure'

 'make'

 'openocd' (to see if its installed now, version 0.10.0+dev-00174-g4befcdc)

(back to the example directory)

 'make flash\_combined' !!(ERROR: LIBUSB\_ERROR\_ACCESS, "no jlink device found")

 'sudo make flash\_combined'

IT WORKS!!!!! The board is not connected and the formware flashed!