KaribeMuriuki
Technology: Learning and Sharing is always fun...

Setting up Linux opensource build and debug tools for freescale freedom board FRDM-KL25Z

By karibe on August 4, 2013

NB: this post has been updated since openocd now implements swd using hidapi and libusb.

Before I dare start reading the mcu manuals, I wanted to just be able to build code, flash and debug on the Freescale FRDM-KL25Z without purchasing software, on Linux. I have read many blogs and here is a summary of what i had to do to get OpenOCD connected to my MKL25Z through SWD.

Step 1: Install open-source toolchain

Install Open-source toolchain(Ubuntu 13.04, 12.04, 10.04) ppa: https://launchpad.net/~terry.guo/+archive/gcc-arm-embedded

For Ubuntu 14.04, you have to download a Linux installation tarball file from launchpad and extract into a favorable directory, like /opt/gcc-arm-embedded/ and jump to step 2.

```
sudo add-apt-repository ppa:terry.guo/gcc-arm-embedded
```

If you are working behind a proxy, make sure you export proxy settings

```
export http_proxy=http://username:password@proxy-server-address:proxy-port
export https_proxy=$http_proxy
export ftp_proxy=$http_proxy
```

and then run add-apt-repository command with -E option

```
sudo -E add-apt-repository ppa:terry.guo/gcc-arm-embedded
```

```
sudo apt-get update
```

…

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

no harm to verify:

```
arm-none-eabi-gcc -v
```

and you should get output similar to:

```
Using built-in specs.
```

```
COLLECT_GCC=arm-none-eabi-gcc
COLLECT_LTO_WRAPPER=/usr/bin/../lib/gcc/arm-none-eabi/4.7.4/lto-wrapper
Target: arm-none-eabi
Configured with:....
```

Step 2: Install pre-requisites:

```
sudo apt-get install libusb-1.0 libusb-dev libudev-dev libtool autoconf automake texinfo build-essential
```

Step 3: Upgrade the kl25z firmware to support CMSIS-DAP

~~You will need a windows system to do this, yes, its due to Companies like freescale that i still have a windows boot in one of my machines, which i never use until they ask me to~~. There is a way to change the firmware on linux. So based on Alan C. assis and Manuelnaranjo, the summary is:

- Grab the Freescale quick start project or the Keil application note
- Extract and locate the file CMSIS_DAP_OpenSDA.s19 or CMSIS-DAP.S19 respectively;
- Put your freedom board in bootloader mode: unplug usb, plug it in while holding reset button, when LED flashes, release the button, you are in bootloader. A 'BOOTLOADER' folder gets mounted on the computer.
- Drag-drop/copy-paste the CMSIS_DAP_OpenSDA.s19/CMSIS_DAP.s19 file into the BOOTLOADER device. wait for a few seconds and unplug it. plug it back in and the LED should go on then off in like a second or so. Now you can go back to Linux :-), lsusb will show the device info.

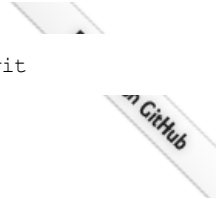
Step 4: install hidapi

Get into the home folder, if thats where you want to have the source:

```
cd ~/
```

Clone the source

```
git clone http://github.com/signal11/hidapi.git
```



Get into the source directory

```
cd hidapi/
```

Run bootstrap

```
./bootstrap
```

configure, generate makefiles

```
./configure
```

compile the sources

```
make
```

Build the binary and install it

```
sudo make install
```

Solve some library location problems

```
sudo ln -s /usr/local/lib/libhidapi-hidraw.so.0 /usr/lib/libhidapi-hidraw.so.0
```

Step 5: Set up udev rules for the debugger

While using hidapi with libusb, hidraw devices are created in /dev as /dev/hidraw2, where 2 is a device number. to have easy permissions settings, add users to the plugdev group and set hidraw rules in udev as follows:

```
sudo nano /etc/udev/rules.d/99-hidraw-permissions.rules
```

and add this line and save by pressing CTL+O (thats an O not a 0) and press ENTER. to exit nano, use CTL+X

```
KERNEL=="hidraw*", SUBSYSTEM=="hidraw", MODE="0664", GROUP="plugdev"
```

No need to restart udev, just unplug and replug the device. hidraw3 will be created with the rw permissions for group plugdev. You can confirm the with the list command

```
ls -l /dev/hidraw*
```

and the output should be something liek this:

```
crw------- 1 root root    251, 0 Jan 28 10:42 /dev/hidraw0
crw------- 1 root root    251, 1 Jan 28 10:42 /dev/hidraw1
crw-rw-r-- 1 root plugdev 251, 2 Jan 28 14:57 /dev/hidraw2
```

Note the use of a wildcard "*" to avoid having to specify the device number, it could be 2 or 3… Like I have written above, use the ls command to confirm the device was generated properly in /dev after you ***Plug In the hardware*** Get out of hidapi source directory

```
cd ../
```

Set up usb device permissions: some people still cant run openOCD successfully without sudo, i figured its permissions needed. use lsusb to see the devices, and locate the Vendor I.D. and Product I.D. of the Keil debugger(c251:f002), then:

```
sudo nano /etc/udev/rules.d/45-keil_debugger.rules
```

Then copy paste this line into that file and save (CTL+O then ENTER) and exit (CTL+X):

```
SUBSYSTEM=="usb", ATTR{idVendor}=="c251", ATTR{idProduct}=="f002", MODE="0660", GROUP="plugdev"
```

Step 6: Install OpenOCD

Clone the repo into a folder openocd/:

```
git clone http://openocd.zylin.com/openocd
```

Change into the repository directory

```
cd openocd
```

Fetch for changes that may not be in mainline openocd: (see comments below)

```
git fetch http://openocd.zylin.com/openocd refs/changes/42/1542/11 && git checkout FETCH_HEAD
```

Run bootstrap script, get all the submodules.

```
./bootstrap
```

The output should be as follows:

```
Bootstrap complete. Quick build instructions:
./configure --enable-maintainer-mode ....
```

Do as bootstrap suggests but with an option to enable cmsis-dap and hidapi-libusb:

```
./configure --enable-maintainer-mode --enable-cmsis-dap --enable-hidapi-libusb
```

Edit the configuration script a bit, attach a halt event

```
gedit tcl/target/kl25.cfg
```

and add the following snippet at the end of that file

```
adapter_khz 50
$_TARGETNAME configure -event gdb-attach {
halt
}
```

Save and close that file

Compile openocd:

```
make -j4
```

Install openocd:

```
sudo make install
```

Run openocd with the right path to the config file

```
openocd -c "interface cmsis-dap" -f /usr/local/share/openocd/scripts/target/kl25.cfg
Open On-Chip Debugger 0.8.0-dev-00331-g1137eae (2014-01-28-12:22)
Licensed under GNU GPL v2 For bug reports, read
```

```
http://openocd.sourceforge.net/doc/doxygen/bugs.html

Info : only one transport option; autoselect 'cmsis-dap'
Info : CMSIS-DAP: SWD Supported Info : CMSIS-DAP: Interface Initialised (SWD)
Info : add flash_bank kinetis kl25.flash cortex_m reset_config sysresetreq adapter speed: 50 kHz
Info : CMSIS-DAP: FW Version = 1.0
Info : SWCLK/TCK = 0 SWDIO/TMS = 1 TDI = 0 TDO = 0 nTRST = 0 nRESET = 1
Info : DAP_SWJ Sequence (reset: 50+ '1' followed by 0)
Info : CMSIS-DAP: Interface ready
Info : clock speed 50 kHz
Info : IDCODE 0x0bc11477
Info : kl25.cpu: hardware has 2 breakpoints, 2 watchpoints
```

Step 7: Install and Configure Eclipse

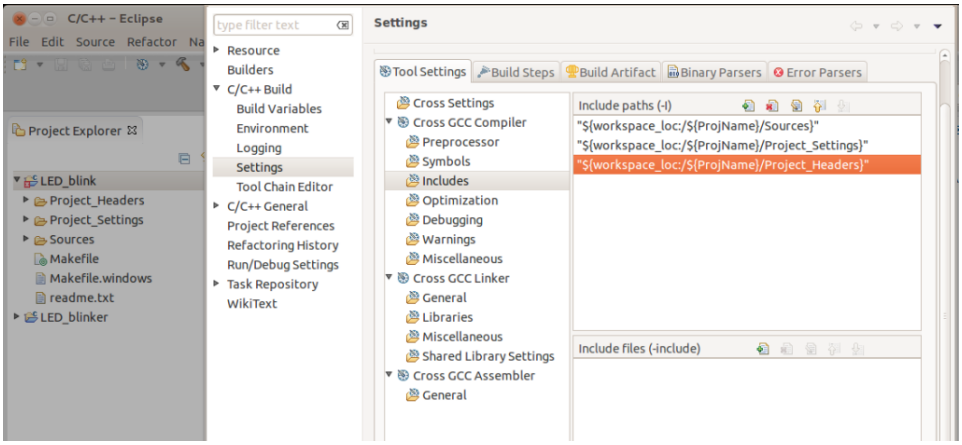
Grab the latest eclipse "Kepler Release" the C/C++ one that comes with CDT plugin. Extract in a working directory like /home/username. Create a workspace like 'mkdir /home/username/frdm-kl25/'. Run eclipse: read this post, then use the workspace created above, select workbench on welcome screen.

In the menu, go to ***help->install new software*** and add this source url: http://download.eclipse.org/tools/cdt/releases/kepler. Install any optional CDT features that were not installed by default, like C/C++ GDB Hardware Debugging. Repeat same procedure for ***GNU-ARM tools***, whose URL is ***http://gnuarmeclipse.sourceforge.net/updates***. Select all options (this now includes support for freescale and ST Microelectronics project Templates) and install. Create a new project: ***File > New > C Project*** Give a ***Project name***, select ***Empty Project*** from the ***Executable*** group and have ***Cross GCC*** as tool chain selected. In the next dialog, unselect ***'Release'*** configuration and leave only ***'Debug'*** For the Cross GCC Command options, set arm-none-eabi- as Cross Compliler Prefix and /usr/bin as ***Cross Compiler Path***, thats where the tools are. Click ***finish***. Now the project is created, there is need to add a starter code implementation, what better than a LED blinker application. Grab Erich Styger's KL25Z_Blink, extract it and then paste the directory tree into the project you created in the workspace. This source has a windows and a Linux Makefile, by renaming the Makefile to Makefile.windows and renaming Makefile.linux to Makefile, and editing the tool-chain location in the Makefile, you can use the installed /usr/bin/arm-none-eabi-gcc to build the project. Just don't use an editor that replaces the tabs with spaces!!
There are 2 ways to build the project. By building ***a Makefile project***, using the already written Makefile or configuring eclipse with the tool-chain and letting it generate the makefiles for you.

To use the custom Makefile, go to ***Project->Properties***, click on ***C/C++ Build***. on ***Builder Settings*** tab, uncheck ***Generate Makefiles automatically*** and set the ***Build directory*** to the path ***${workspace_loc:/Project_Name/}*** where Project_Name is the name of the project you created.
You can now click build button and generate the executables.

To configure eclipse to generate makefiles, Select the project in the 'Project Explorer' and go to ***Project->Properties***, click on ***C/C++ Build***. On ***Builder Settings*** tab, check ***Generate Makefiles automatically***. On ***C/C++ Build->Settings***, in the 'tool settings' tab, under ***Cross GCC Compliler->Includes***, set the include folders by clicking on the + sign and browsing the workspace at the bottom of the pop-up window, to the project and the source folder.

It should look something like this:



eclipse-includes-all

If you are using processor expert, you have to add these two folders in the includes:

1. path-to-eclipse/ProcessorExpert/lib/Kinetis/iofiles
2. path-to-eclipse/ProcessorExpert/lib/Kinetis/pdd/inc

Under ***Cross GCC Compliler->Miscellaneous***, add this to the default settings

```
-mcpu=cortex-m0 -mabi=aapcs -mthumb -msoft-float
```

Under the ***Cross GCC Linker->General***, check 'do not use standard start up files'.

Under ***Cross GCC Linker->Miscellaneous*** specify the mthumb option by the linker flags:

```
-mthumb -mcpu=cortex-m0 -T${ProjDirPath}/Project_Settings/Linker_Files/MKL25Z128_flash.ld
```

NB: In case you try a build and edit settings and you still got errors, try rebuilding the index under

Project->C/C++ Index-> Rebuild Build the project and you should have no errors

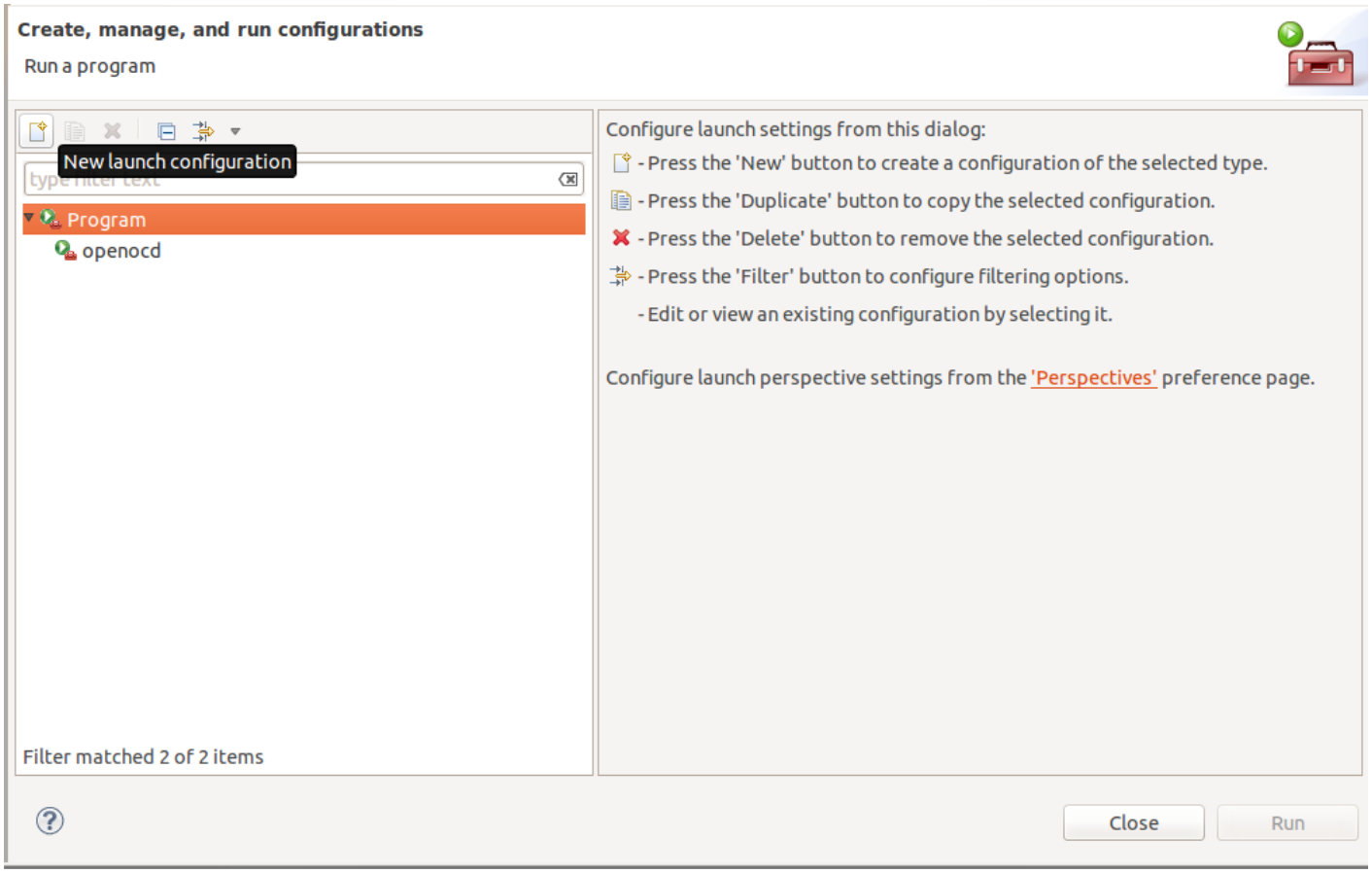
Integrate OpenOCD and arm-none-eabi-gdb with Eclipse

This is done using external tools, as outlined by Shanjit Singh Jajmann.

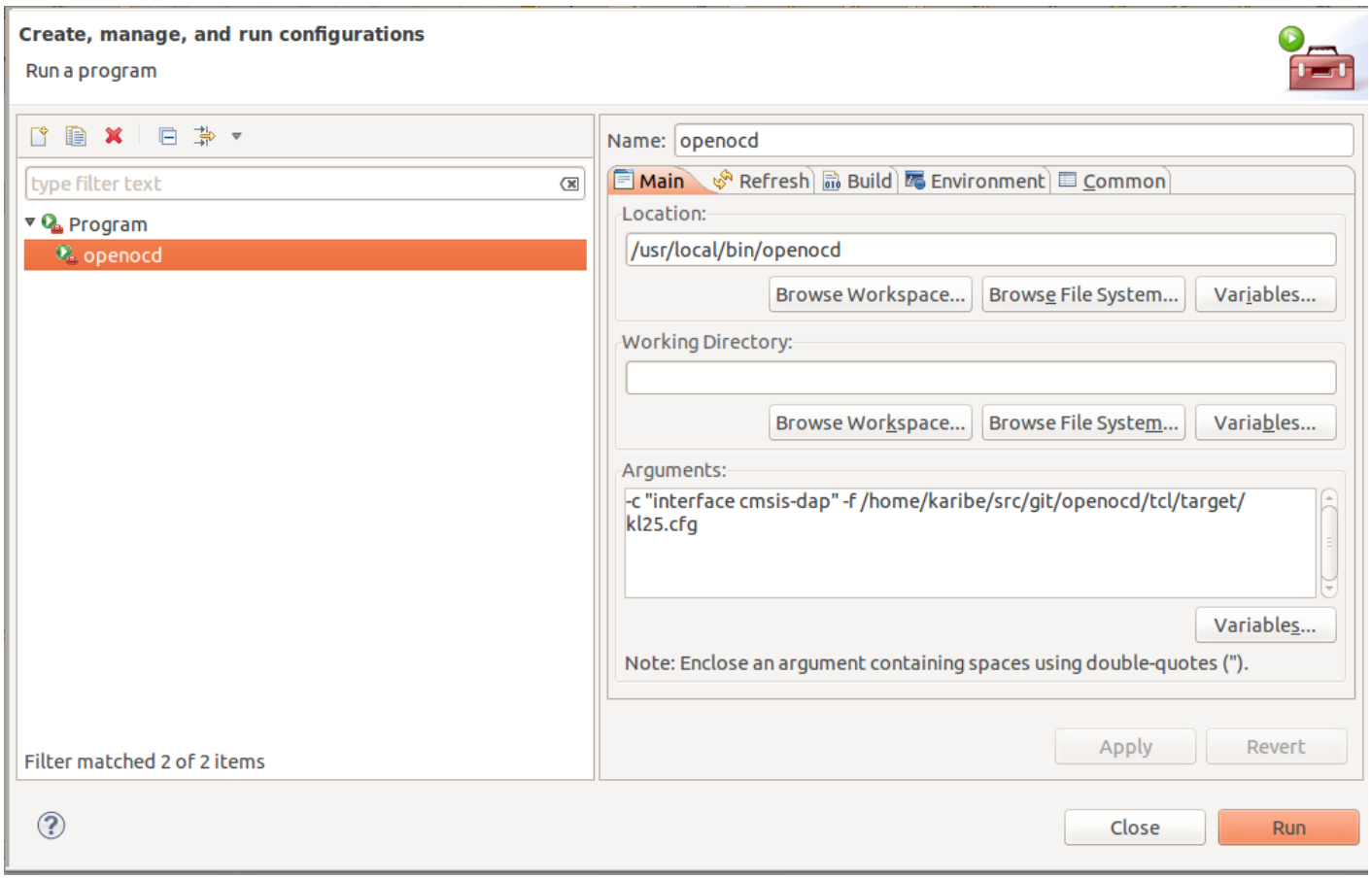
Go to ***Run->Debug->External tools*** and highlight Program, then create new:

Enter the path to openocd and the arguments used in terminal launch:

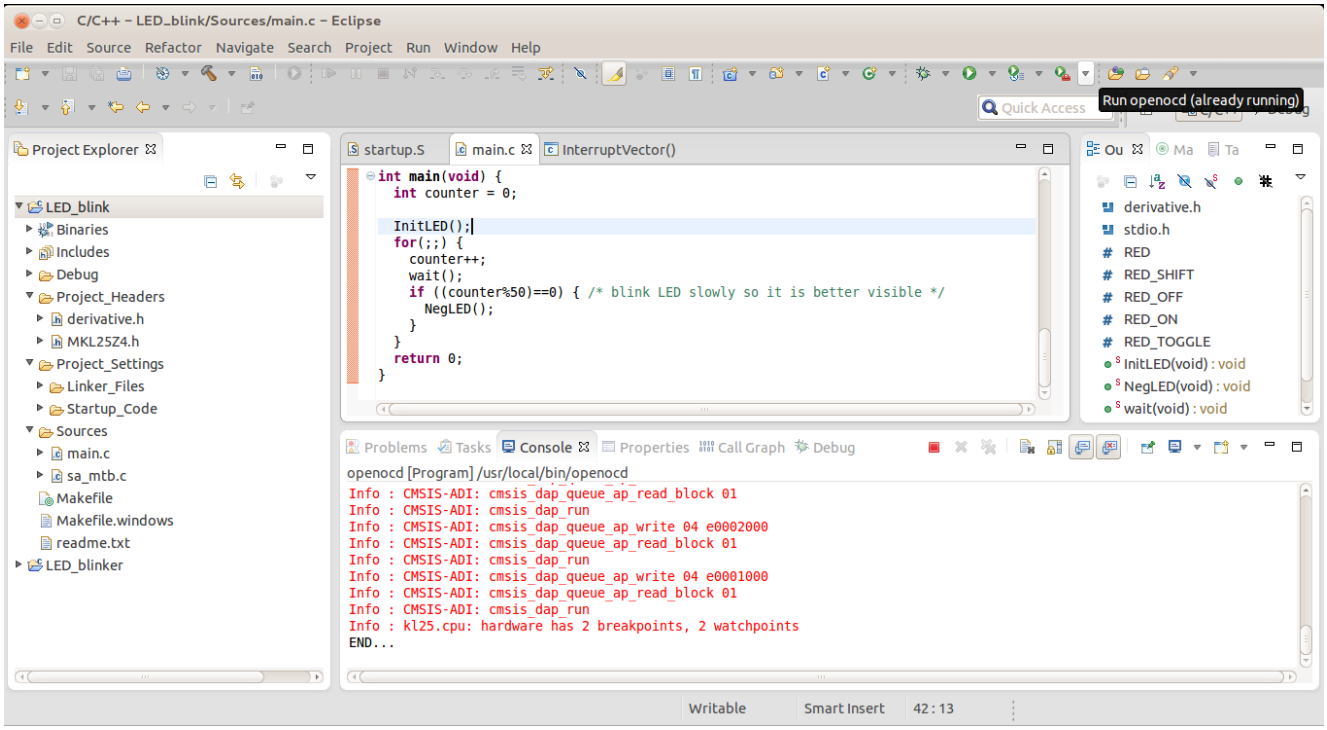
You can go to the common tab and check external tools under ***Display in favorites menu***. This adds an ***external tools*** icon in the IDE menu, you will use that to start a debug session.



eclipse external tools new instance



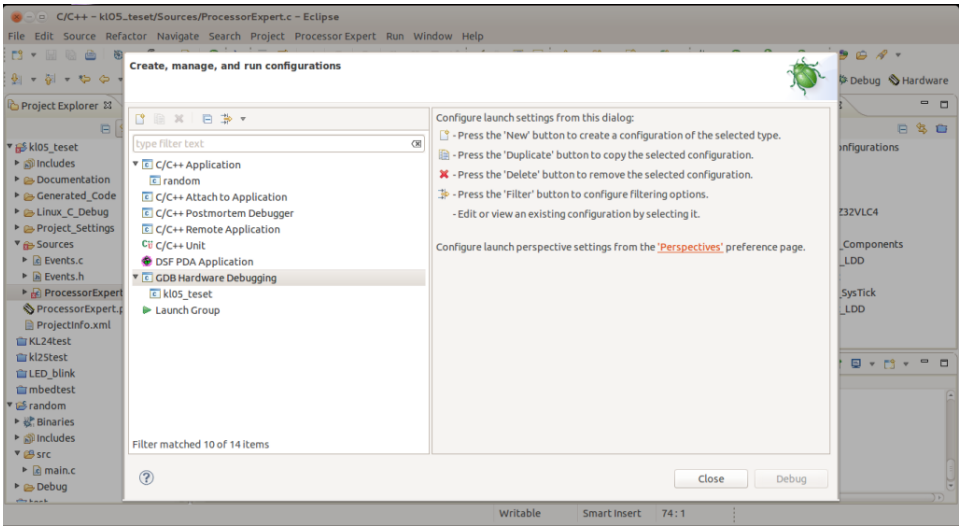
openocd eclipse external tools



eclipse openocd integration

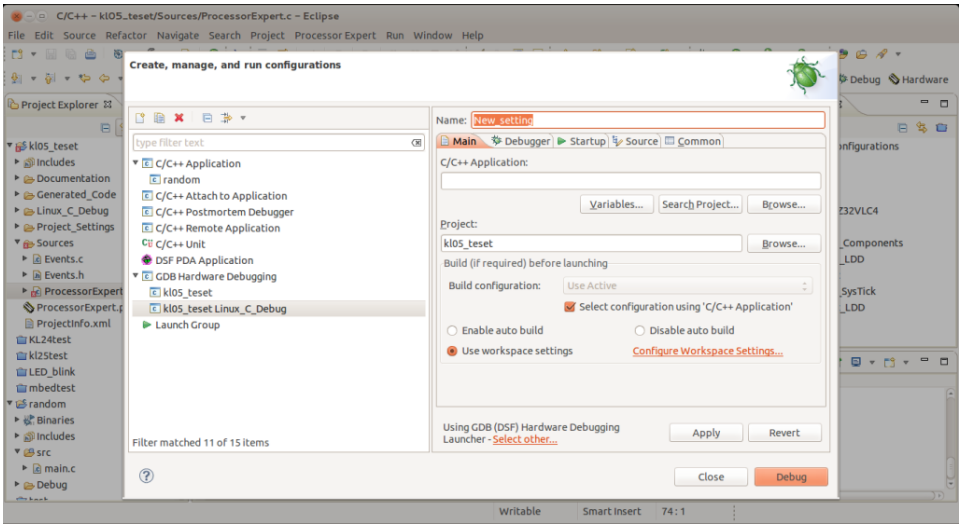
Starting a debug session:

To debug your target, you need to configure the debugger client in a way that it will connect to the debug server(OpenOCD). To do this, click on the down arrow next to the bug looking icon to get a drop-down list and click on ***Debug Configuration*** this window pops up:



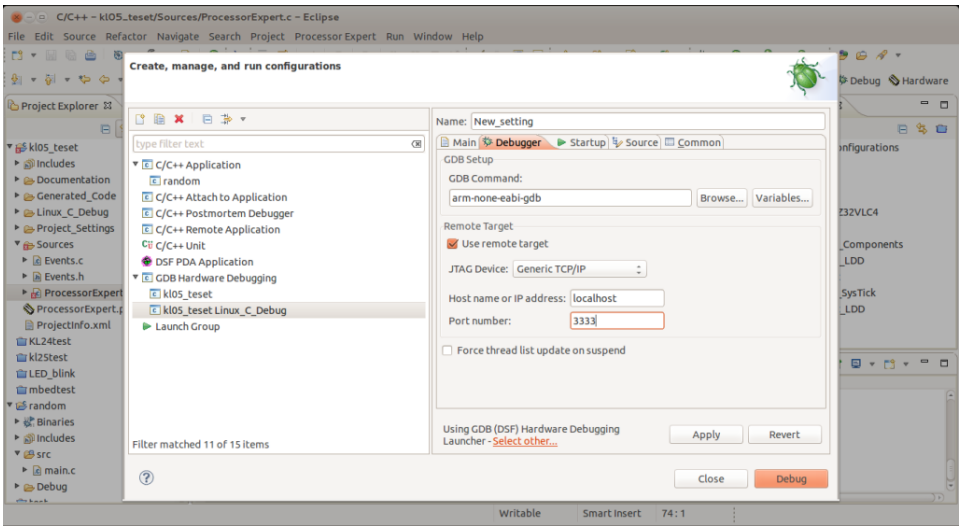
New config pop-up

Next, click on ***GDB Hardware Debugging*** and click on the 'new entry'(like a new file) icon at the top left corner of the popup: enter or change the name field, in this case, I entered "New_setting".



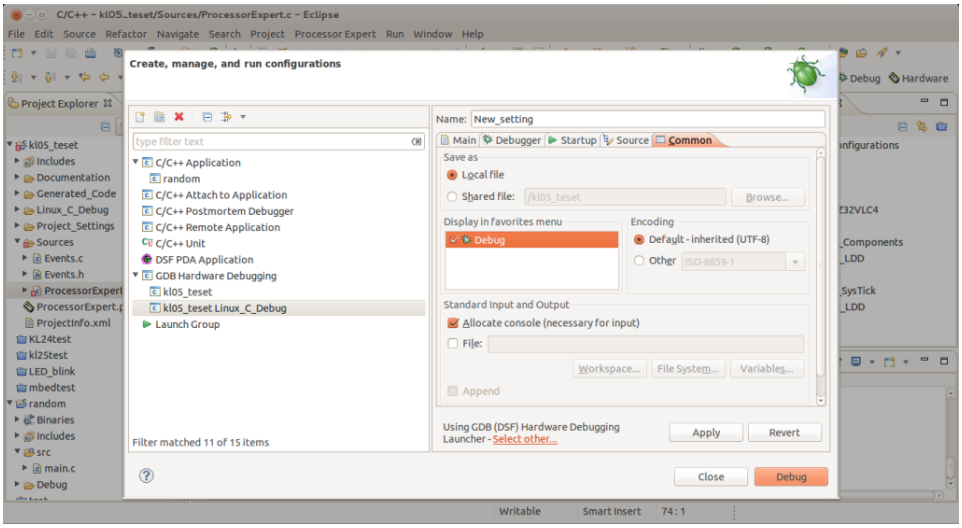
New_debug_setting

Go to the debugger tab on the right, enter the name of debugger as GDB Command, that is “arm-none-eabi-gdb”. I guess the system will get it in PATH, in case not, enter with full path, i.e /usr/bin/arm-none-eabi-gdb, if that is where it is :). Under Remote Target, check ***use remote target***, choose a ***Generic TCP Connection*** from the drop-down and set port to 3333 for OpenOCD serves through that port.



gdb_connection_config_tab

The last thing is about making it easy to find this project’s debug settings within a click. Go to the ***common*** tab. Under ***dislay in favorites menu***, check the checkbox. Click apply and then debug if you want to debug right away, or close if you want to debug later, who knows?



add_debug_config_to_favorites

References:

1. Freescale quick start project
2. Keil application note
3. Manuelnaranjo
4. Alan C. assis:kl25z firmware
5. Alan C. assis: openocd
6. Eclipse kepler
7. MCUoneclipse
8. Shanjit Singh Jajmann on github

UPDATE:

There was a problem with “reset” instruction in gdb with the cmsis-dap interface for kl25z. This was openocd implementation problem which has since been solved.

Debugging now fully working, see comments down below…

The setup gives two hardware breakpoints, so if you put more than two, you will end up with the resource not found situation. otherwise everything runs ok for the FRDM-KL25Z and FRDM-KL05Z boards as well as with external targets. I tested with a Processor Expert project toggling RGB LED.

Posted in Embedded systems, Linux | Tagged ARM, build, eclipse, embedded systems, Freescale, microcontroller, Processor Expert, ubuntu | 52 Responses

Articles in this series

- Debugging arm freescale microcontrollers with J-Link GDB Server and GNU-ARM toolchain gdb with semihosting in Linux

- USBDM ARM debug Server for Freescale micro-controllers in Linux with GNU Debugger
- TWR-K60N512 full debug with OpenOCD and GNU-ARM in Linux
- kl05z kinetis mcu custom circuit and Processor Expert Valentine LED twinkle
- Freescale TWR-K60N512 tower kit SWD debug with FRDM-KL25Z CMSIS-DAP firmware
- custom pcb board for the freescale kl24z QFN chip
- Processor Expert in Eclipse Generating Code for Kinetis L-series Microcontrollers
- Building mbed frdm-kl25z C++ applications with eclipse in linux
- Online mbed compiler and the freescale freedom board frdm-kl25z LED_blinker
- Programming the FRDM-KL25Z board with a bare-metal project
- Setting up Linux opensource build and debug tools for freescale freedom board FRDM-KL25Z

52 responses to “Setting up Linux opensource build and debug tools for freescale freedom board FRDM-KL25Z”

DIY Free Toolchain for Kinetis: Part 5 – Linux Host with OpenOCD and CMSIS-DAP | MCU on Eclipse August 5, 2013 at 2:56 pm | Permalink

> […] Setting up Linux opensource build and debug tools for freescale freedom board FRDM-KL25Z […]

Jorge Hernandez August 6, 2013 at 7:22 am | Permalink | Reply

> When i run make, i get an error that says:
>
> cmsis_dap.c: In function ‘cmsis_dap_cmd_DAP_SWJ_Clock':
> cmsis_dap.c:274:50: error: declaration of ‘clock’ shadows a global declaration [-Werror=shadow]
> cc1: all warnings being treated as errors
>
> how can i fix this? why is this happening? (i’m a noob regarding “make”). Thank you Karibe. Keep up the good work!

karibe August 7, 2013 at 12:33 pm | Permalink | Reply

> Its a werror as shown, c++ warnings taken as errors, its simple to disable during configuration. It was mentioned in this comment
> Use this to configure:
> ./configure –enable-maintainer-mode –enable-cmsis-dap –disable-werror

---

Programming the FRDM-KL25Z board with a baremetal LED toggle >> Karibe August 17, 2013 at 12:59 am | Permalink

[…] this post on how to setup arm-gnu […]

Online mbed compiler and the freescale freedom board frdm-kl25z LED_blinker >> Karibe August 21, 2013 at 3:07 am | Permalink

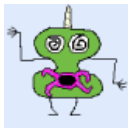
[…] version 1.08, you can get it from pemicro. Look for the BOOTUPDATEAPP.SDA. follow the steps in this post to update […]

MB September 8, 2013 at 7:51 pm | Permalink | Reply

Well Done. Thanks for documenting it. This will be useful for so many people. I don't know why element14 and freescale don't push more with open source tools.

karibe September 14, 2013 at 2:01 am | Permalink | Reply

Thanks MB, i wonder the same too. I wish openocd works fully as a debugger for this kl boards, but its getting there. Lets hope the devs will get us there, its not a priority for Freescale you know :). Ah, the dev's already did, see Hagai Cohen's comment below!!



allankliu September 12, 2013 at 12:41 pm | Permalink | Reply

I am trying my best to support KL25Z with Wiring API, like Arduino did. Before porting API, I have to setup a complete open source toolchain on Windows.

The C version built with arm-gcc-embedded runs OK on FRDM. But CPP version didn't. I have no idea where I have done wrong.

Do you have any working demo? which can be built with arm-gcc-embedded and makefile? Maybe I can find my problem after cross-checking the makefile.

Or, should I run the code in mBed and export to arm-gcc-embedded?

karibe September 13, 2013 at 9:07 pm | Permalink | Reply

> First of all, windows and opensource?… I do not use windows and am no much help there. secondly, in this post I described how to set up an offline version of mbed and generate the library to be built with arm-gcc-embedded there is a Makefile in there, as well as a sample application, am not sure if that will help. Tell me if it doesn't. I actually for a moment there thought this comment was on that post 😀.

Hagai Cohen September 17, 2013 at 10:56 am | Permalink | Reply

I Was able to make it work!

the openocd cmsis-dap support are at last stages of a approval, so at this moment they are not approved yet. probably later on they will be, and be merged into mainline.

I was using http://openocd.zylin.com/#/c/1542/11.

1.Clone to OpenOCD Repo:

git clone http://openocd.zylin.com/openocd cd openocd

2.Since it's not merged into mainline, i took the patch:

git fetch http://openocd.zylin.com/openocd refs/changes/42/1542/11 && git checkout FETCH_HEAD

3.bootstrap & configure:

./bootstrap ./configure –enable-maintainer-mode –enable-cmsis-dap –enable-hidapi-libusb

4.added adapter settings into kl25.cfg (hopefuly this will be added into the mainline as well.)

vim ./tcl/drivers/kl25.cfg

```
adapter_khz 50
$_TARGETNAME configure -event gdb-attach {
halt
}
```

5.make & sudo make install

6.setup UDEV Rules. (in the post)

7.test it works:

openocd -c "interface cmsis-dap" -f /usr/local/share/openocd/scripts/target/kl25.cfg

karibe September 17, 2013 at 9:10 pm | Permalink | Reply

Thanks Hagai. This is very good news, Its been a while since i checked progress on this. I will try to reproduce your success with the kl25 board, that will be really awesome. Thank you for taking the trouble to post a comment 😃

karibe September 19, 2013 at 2:18 am | Permalink | Reply

> Ok, for *hidapi* in ubuntu, you have to install libudev and this is done by running the command *sudo apt-get install libudev-dev*. I also didn't understand the path *./tcl/drivers/* so i used *nano tcl/target/kl25.cfg* and added the configurations at the end of the file. Take care for the double dash "- -" characters in config options, they are actually rendered differently in html, retype them in terminal, anyway, everyone knows an option is identified as *–enable-maintainer-mode* so edit after copy-pasting.

Sándor Bognár September 23, 2013 at 10:43 am | Permalink | Reply

Hi!

I also managed to debug kl25 apps by openocd, but I missed the uart terminal
badly, it is not too fast and still in beta state.
I installed the USBDM and it works perfectly. Having a very fast flash, debug
and uart interface.
I is more than perfect.

I work on SuSe linux 12.3 and there is no 64 rpm package for it, that's why I had to compiled the git version.

Cheers,

Sándor

karibe September 24, 2013 at 9:03 am | Permalink | Reply

> Hi Sándor, I also heard about USBDM, I cant remember why I never got through with it, I will check, always looking for alternative Linux tools. Thank you, I will look at building from git source and see if i can confirm this myself. For the terminal, you can have a serial terminal in eclipse. look at the first answer to this stackoverflow question, I installed it in Kepler release easily 😃

karibe October 4, 2013 at 11:35 am | Permalink | Reply

What git repo did you build USBDM from? if you dont mind sharing a link…

Ed October 30, 2013 at 7:57 pm | Permalink | Reply

Did exactly all steps and am getting this, altho i did made some misstakes at first, maybe i need to uninstall and redo, but how?

~/openocd$ openocd -c “interface cmsis-dap” -f tcl/target/kl25.cfg
Open On-Chip Debugger 0.7.0-dev-g1d8f0c1 (2013-10-27-21:43)
Licensed under GNU GPL v2
For bug reports, read
http://openocd.sourceforge.net/doc/doxygen/bugs.html
Info : only one transport option; autoselect ‘cmsis-dap’
Info : CMSIS-ADI: cmsis_dap_select
Info : CMSIS-DAP: cmsis_dap_swd_init
Error: Can’t find CMSIS-DAP Interface! Please check connection and permissions.
Error: Error selecting ‘cmsis-dap’ as transport
in procedure ‘interface’

karibe October 31, 2013 at 11:21 am | Permalink | Reply

from the post, “lsusb will show the device info you can use to write udev rule, or opt to run openocd with sudo…” read carefully. you can set up permissions by writing a udev rule, or run that command with sudo.

Marcos November 4, 2013 at 1:04 pm | Permalink | Reply

Hi Karibe

Did you find any solution to **the invalid command name “jtag”**?

karibe November 6, 2013 at 1:11 am | Permalink | Reply

yes, its in the comments above

Patrick Krekelberg December 5, 2013 at 10:24 pm | Permalink | Reply

Hi Karibe!

Really nice work on this. I know how much effort it takes to hand-roll a toolchain, let alone spend the time to write about it so others can benefit. Thanks!

I'd been trying to get this going for OS X. I was unable to build OpenOCD due to some environment issues. Have you heard when the CMSIS-DAP support will go into the OpenOCD mainline? I looked at the code review page and it appeared to be still in process.

My only reservation about all this is that the Processor Expert component does not function on OS X. I am honestly unsure whether I'll need it or not, but it's nice to have the option. What are your thoughts?

MCU vendors seem very pro-Windows unfortunately. Anything that allows me to stay in unix land is well worth some extra effort, but I still haven't found a way toward a full development/debugging/production workflow for Freescale KL25/KL46 on mac/linux.

As a backup I am spinning everything up with bare Eclipse in a Windows VM as well (thanks to Erich's writings at http://mcuoneclipse.com)…

Cheers,
Patrick

karibe March 4, 2014 at 11:01 am | Permalink | Reply

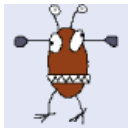
Hi Patrick, I missed this comment. By now you must have realized things have really changed, we can have full debug in Linux. I dont know about mac, but i guess you can easily find that out or port the tools from Linux.

Freescale freedom board FRDM-KL05Z with openocd and GNU-ARM tools | Karibe January 28, 2014 at 1:56 pm | Permalink

[…] for Freescale mcus (and other arm-based mcus) using open-source tools as documented on this post. One of my students had a look at the FRDM-KL05 and it can actually be used in exactly the same […]

ARM microcontroller emulation using QEMU | Karibe February 1, 2014 at 2:59 pm | Permalink

[…] I already covered installation procedures for gnu-arm toolchain in this post. […]



Nick February 7, 2014 at 7:18 am | Permalink | Reply

Hi Karibe,
Thanks for posting this guide. When I try your steps to install hidapi, it fails when after I download the files from github and try to run ./bootstrap, it says "autoreconf: not found"
Do you know how I'd get autoreconf set up so that this will work? I'm running ubuntu 13.10 64 bit.

karibe February 7, 2014 at 2:57 pm | Permalink | Reply

have you installed auto-tools? you need to install all pre-requisites

Rafael Murakami February 17, 2014 at 1:05 pm | Permalink | Reply

Karibe Hello, congratulations on the blog! I did the whole process, but I have a doubt. After running opencd what the process required to compile the code for my board?

karibe February 19, 2014 at 10:24 am | Permalink | Reply

Click "build", if you cant find that icon i suggest you learn some eclipse basics, try eclipse help menu.

Rafael Murakami February 20, 2014 at 8:31 pm | Permalink | Reply

Yes, the binary code and .s19 were generated, but dont compiled for my board. Opencd its running with message 2 breakpoints and 2 watch points, but dont appear te END message like your print…

karibe February 20, 2014 at 9:16 pm | Permalink | Reply

If you set up properly, the fact that you have binaries means you have build the project for your board. Now you need to flash the binary into the board using gdb, set up debug configurations like i have added

at the end of the post. then add break points and press F8 to continue or F5 for single stepping. I hope the screenshots are close enough to help you set up the debug interface.

Rafael Murakami February 22, 2014 at 10:16 pm | Permalink | Reply

Thanks Karibe! I got compile the code to the board. Now I will learn to work with this interface …

Alex Makau February 26, 2014 at 12:13 pm | Permalink | Reply

when installing hidapi i reached the "/cofigure" step but t=i am getting the below problem but i seem not to find a way to install libudev.help

checking for libudev… no

Library libudev was not found on this system.
Please install it and re-run ./configure

karibe February 26, 2014 at 10:12 pm | Permalink | Reply

I restructured the post, moved the prerequisites to the beginning 😃

kevin March 4, 2014 at 8:40 pm | Permalink | Reply

run….

sudo apt-get install libudev-dev

then retry….it will work av tried it out n its working

karibe March 7, 2014 at 9:00 pm | Permalink | Reply

Yeah Kevin, you are right. I added libudev-dev to the list of prerequisites at the beginning of the post. I did not need to install it for the procedure because I was already using it for other stuff in my installation at work.

kevin March 4, 2014 at 8:36 pm | Permalink | Reply

just before installing OPENOCD am stuck at this stage where am supposed 2 check

ls -l /dev/hidraw*

is ok and the files are present…….but av found out the file r not being created…even checked manually …and repeated previous stages again….giving me

nivek@nivek-HP-Pavilion-dv6-Notebook-PC:~$ ls -l /dev/hidraw*
ls: cannot access /dev/hidraw*: No such file or directory

karibe March 5, 2014 at 1:15 am | Permalink | Reply

The hidrawX device, where X is the device number is created when you plug in the hardware

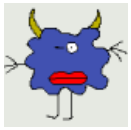
Alex Makau March 5, 2014 at 12:57 pm | Permalink | Reply

while installing OPenOCD am getting the following error when i run configure.
configure: error: /bin/bash ./config.sub …. failed

help out

karibe March 7, 2014 at 8:48 pm | Permalink | Reply

At what point? I think we solved this one somehow, you had not run ./configure properly, or… may be you are passing a parameter wrongly to the ./configure script, paste the command you are using here we see.



Iker March 13, 2014 at 3:48 pm | Permalink | Reply

Hello, very nice tutorial! I am trying to use the KL46 from freescale. I have still a problem connecting to the device. After drag'n dropping the CMSIS SDA on the board in BOOTLOADER mode I unplug and plug again the device. No drive appears then. I try anyway to go on in the tutorial but after running sudo openocd -c "interface cmsis-dap" -f etc… in the end I get:

Polling target kl46.cpu failed, GDB will be halted.
Error: CMSIS-DAP: Read Error (0x04)

Any help will be appreciated.

Thanks

karibe March 14, 2014 at 12:02 pm | Permalink | Reply

Is it a KIT or a custom board? Changing the firmware has to be done in a windows environment. Post the commands you ran and the output immediately after that command. can you lsusb and see the device? if so, post the .cfg file you are using to connect. It is also better if you set up permissions instead of using sudo command. From what I can see, I am imagining it connected and then disconnected, mostly problems with custom boards because of the reset oscillating, before the chip is programmed for the first time, if not so, because of the watchdog being enabled by default. Try USBDM with usbdm firmware if debugging a custom board, worked best for me.

Developing for FRDM-KL25Z platform in Linux – Flashing | CoolParadox Blog March 14, 2014 at 7:07 am | Permalink

[…] Setting up Linux opensource build and debug tools for freescale freedom board FRDM-KL25Z […]

Desenvolvendo em Linux para FRDM-KL25Z – Gravação | CoolParadox Blog March 14, 2014 at 7:50 am | Permalink

[…] Setting up Linux opensource build and debug tools for freescale freedom board FRDM-KL25Z […]

tobias March 17, 2014 at 7:25 pm | Permalink | Reply

i have this error

program “arm-non-eabi-” not found in path

is this a problem with the path i cant seem to remove the error…help

karibe March 18, 2014 at 3:30 am | Permalink | Reply

Hi Tobias, Check your **Tool Settings** under **C/C++ Build->Settings**, **Cross Settings**->**Prefix=arm-none-eabi-**, NOT **arm-non-eabi-**, Path=/usr/bin and under Cross GCC Compiler->Command=gcc. If those are combined, you will end up calling $Path$prefix$Command which translates to /usr/bin/arm-none-eabi-gcc. I hope that helps.

---

kl05z kinetis mcu custom circuit and Processor Expert Valentine LED twinkle >> Karibe March 18, 2014 at 9:34 pm | Permalink

[…] A GUI pops up, it will automatically detect the board as a BDM device, click on the TARGET tab and click on Detect chip to auto-detect the target chip, after that, set the port to serve through under GDBServer (you will use this in the eclipse debug configuration) and click on Keep Changes. A terminal window shows up and you are good to go. the rest of the debug settings remain as usual like OpenOCD settings in external tools, see the post on setting up debug tools in Linux […]

USBDM ARM debug Server for Freescale micro-controllers in Linux with GNU Debugger >> Karibe March 19, 2014 at 12:43 am | Permalink

[…] A GUI pops up, it will automatically detect the board as a BDM device, click on the TARGET tab and click on Detect chip to auto-detect the target chip, after that, set the port to serve through under GDBServer (you will use this in the eclipse debug configuration) and click on Keep Changes. A terminal window shows up and you are good to go. the rest of the debug settings remain as usual like OpenOCD settings in external tools, see the post on setting up debug tools in Linux […]

kevin March 22, 2014 at 7:58 pm | Permalink | Reply

plz remind me what we were editing to avoid this error when setting “sudo add-apt-repository ppa:terry.guo/gcc-arm-embedded” am doing the whole process again…..

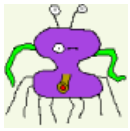
Cannot add PPA: ‘ppa:terry.guo/gcc-arm-embedded’.
Please check that the PPA name or format is correct.

karibe March 23, 2014 at 10:44 pm | Permalink | Reply

We were exporting proxy settings and using “sudo -E add-apt repository “

Debugging arm freescale microcontrollers with J-Link GDB Server and GNU-ARM toolchain gdb with semihosting in Linux >> Karibe March 29, 2014 at 9:01 pm | Permalink

[…] I am assuming you have done the first three, if not, gather these details on this post […]



Pinkesh October 20, 2014 at 9:20 am | Permalink | Reply

Hello Mr.Karibe

Thanks for your stuff.
I installed Openocd the way you mentioned here . But I am getting following erro while flashing.

Error: flash driver ‘at91samd’ not found
Runtime Error: /usr/local/share/openocd/scripts/target/at91samdXX.cfg:60:
in procedure ‘script’
at file “embedded:startup.tcl”, line 58
at file “/usr/local/share/openocd/scripts/board/atmel_samr21_xplained_pro.cfg”, line 8
in procedure ‘flash’ called at file “/usr/local/share/openocd/scripts/target/at91samdXX.cfg”, line 60
make: *** [flash] Error 1

karibe October 21, 2014 at 12:47 am | Permalink | Reply

The error means you are missing the flash driver. I have never used an atmel board with openOCD but, may be its esier to check what’s up if you paste the config files you are using somewhere online and share some links here. that way we can look at what you have as the configuration.

Hemant Joshi January 30, 2015 at 6:23 pm | Permalink | Reply

I am using FRDM-KL25Z board with mbed. Hence installed “20140530_k20dx128_kl25z_if_opensda” firmware provided by Freescale. Since mbed online compiler/linker doesn’t allow me to generate a CRP (Code Read Protection) enabled binary, I hand modified the binary so that I could change value at 0x0000040C. However the OpenSDA FW doesn’t allow me to flash the file, giving error report “Reserved Bits”. I want to use mbed but want OpenSDA to ignore the modification. One way to do so is to modify the CMSIS-DAP source code and remove the check and rebuild. But before doing so, I wish to know if there is any simpler way or alternative to flash my modified mbed generated bin file?
At the same time I strongly wish to use mbed.

Thanks in advance,
-Hemant

Leave a Reply

Name *

Email *

Website

Math Captcha *
seventy nine + = eighty four

Comment

Post Comment

« Previous Next