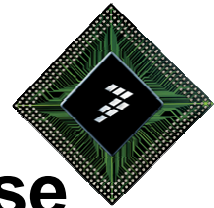




Alternative tool for i.MX JTAG debugging How to use OpenOCD + JTAG probe + Eclipse



Alan Assis, Renato Frias and Rogerio Pimentel

Goal

The aim of this work is to provide directions to setup a low cost development environment option to bring up i.MX designs.

This presentation does not provide details about the tools, it rather shows how to build and use them. A few Internet links are provided for those interested on more details about the tools.



Tools - What is needed?







- **ARM toolchain**
 - Included on FSL BSP.
- **OpenOCD**
 - Open Source project.
- **Eclipse IDE**
 - Open Source project.
- **Jtag Probe**
 - Several options available, including build your own.
- **Hardware Prototype**
 - Target board to be debugged.



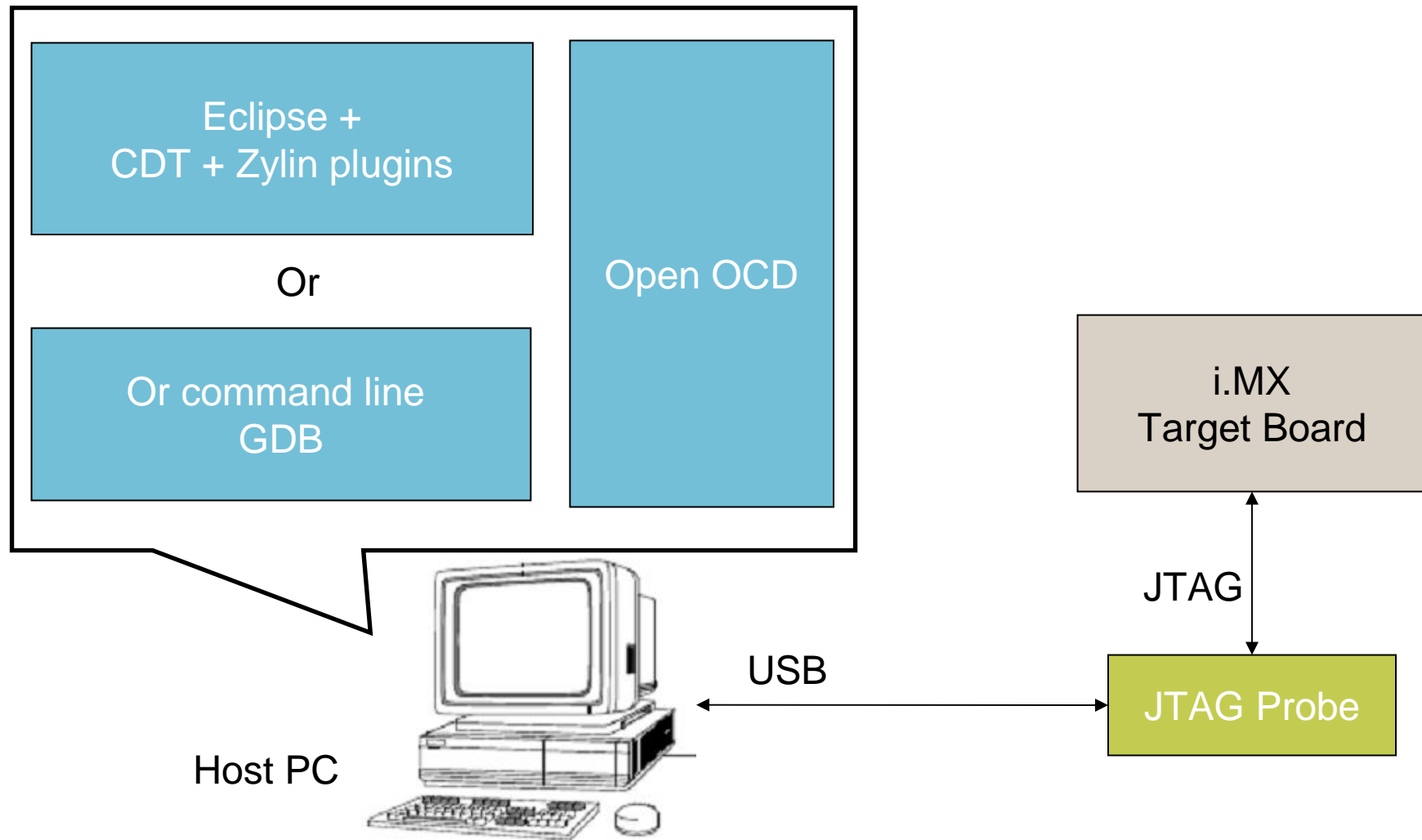
Warning: The next pages will detail the toolset install procedure for Linux hosts, but it is possible to setup the environment for Windows hosts with a few changes.

Low Cost Debug Options

- My new i.MX board is ready, how do I debug it?
 - Besides blinking LEDs... 😊

Project Phase Vs. Debug Tool	Bring up	Bootloader	OS / Apps
JTAG			
Serial / Eth			

Development Tool Overview





Using the tools

A first debug session – Blinking LEDs

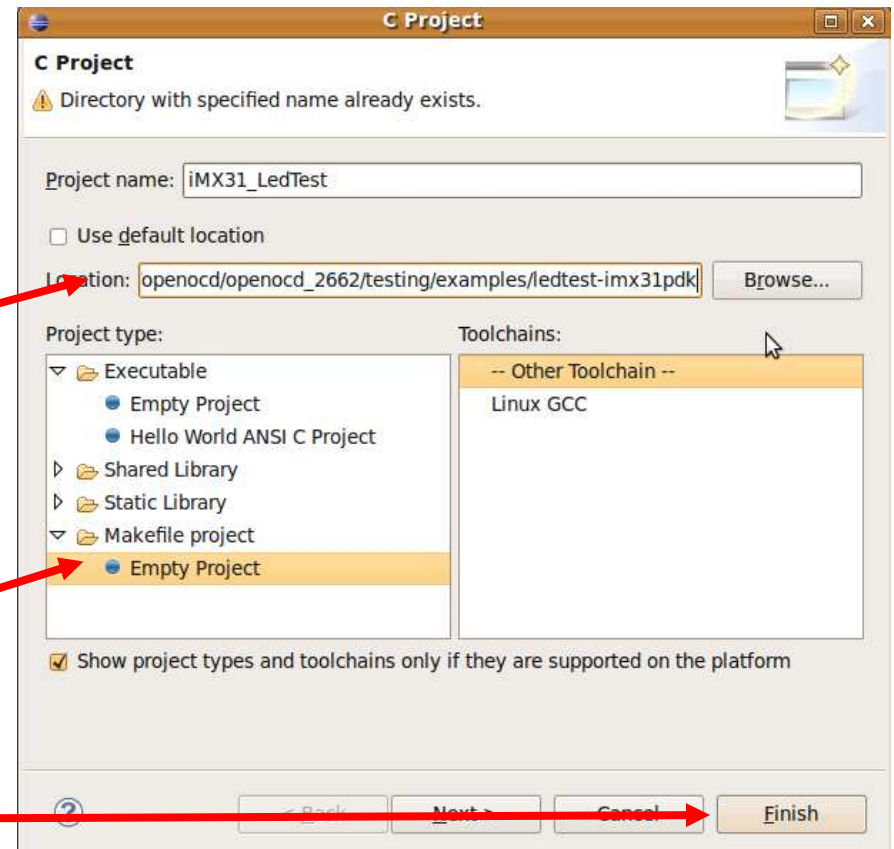
Alan Assis included some example test code on Open OCD for i.MX27 and i.MX31. We can follow the steps below to run a simple environment check.

1. Setup Eclipse Project.
2. Turn on the target board on bootstrap mode.
3. Start OpenOCD.
4. Launch Debugger on Eclipse.
5. Debug.

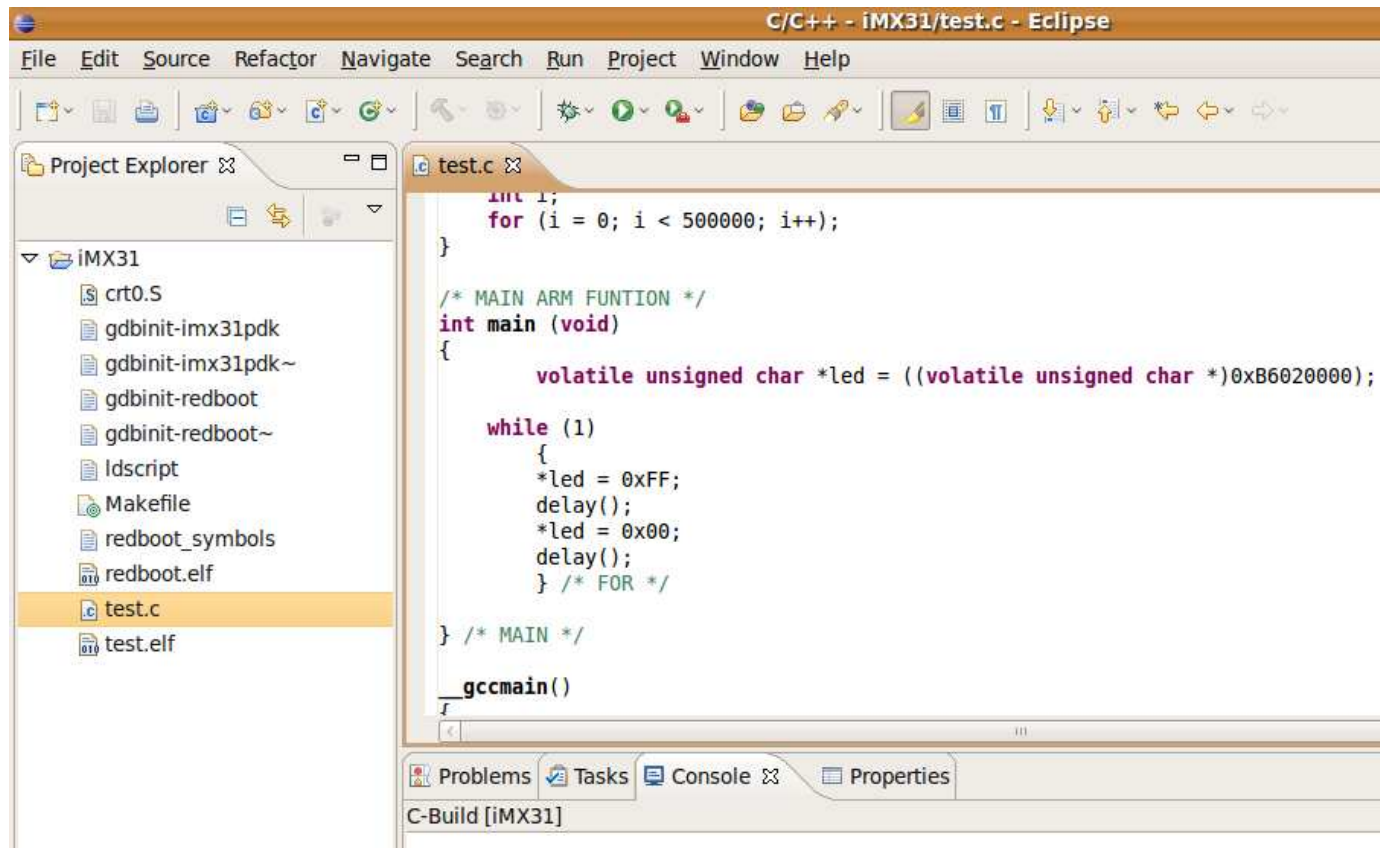


A first debug session – Blinking LEDs

1. Start Eclipse Project. (Optional)
 - Start Eclipse and click on New -> C Project
 - Use the “<openocd path>/testing/examples/ledtest-imx31pdk” as location.
 - Chose Makefile project.
 - Click Finish.



A first debug session – Blinking LEDs



2. Turn on the board on bootstrap boot mode

A first debug session – Blinking LEDs

3. Start OpenOCD.

```
$ sudo openocd -f interface/signalyzer.cfg -f board/imx31pdk.cfg
```

File with jtag probe details.

File describing board specific configuration.

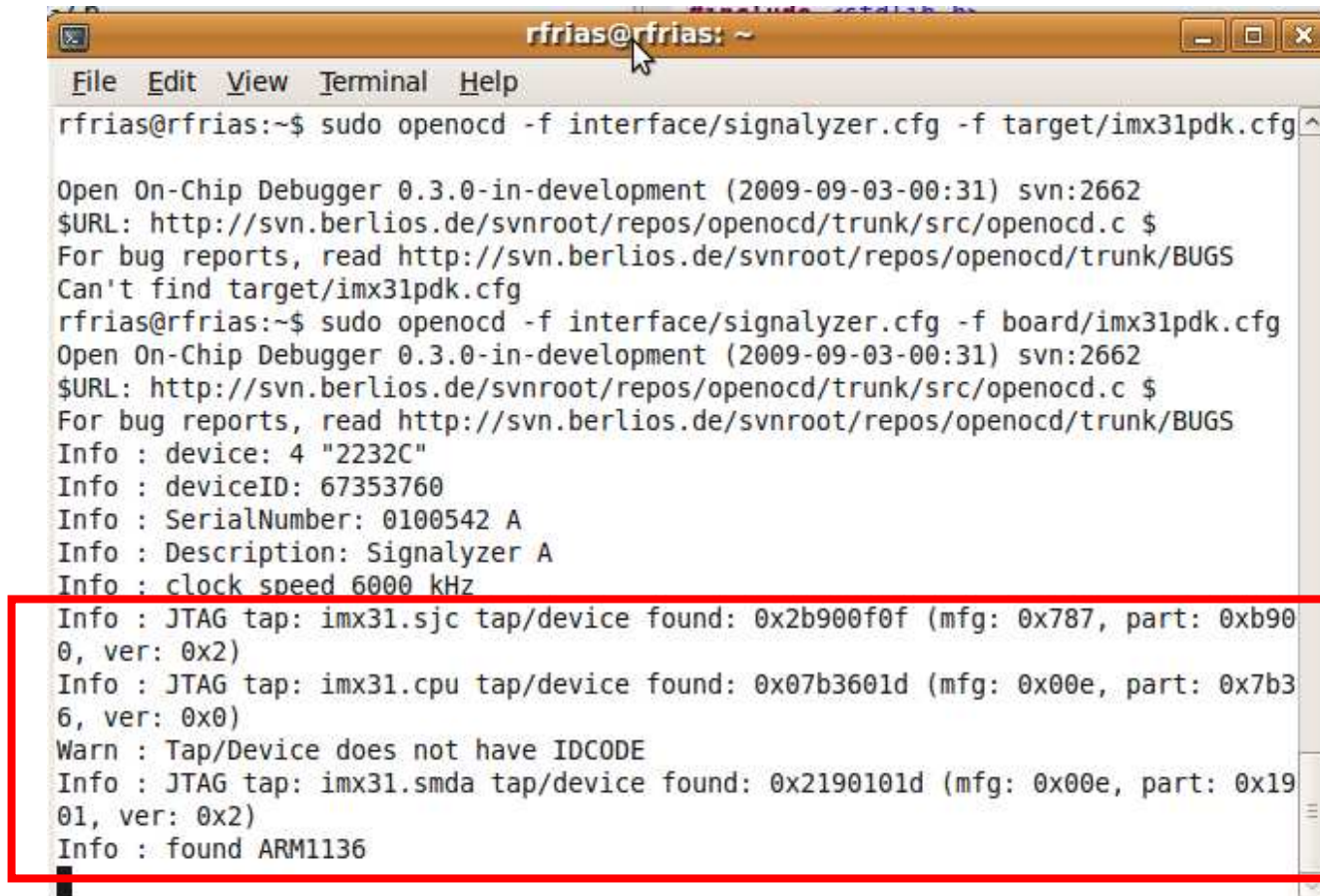
```
#interface
interface ft2232
ft2232_device_desc "Signalyzer A"
ft2232_layout signalyzer
ft2232_vid_pid 0x0403 0xbca0
```

```
# =====
# Init CCM
# =====
mww 0x53FC0000 0x040
mww 0x53F80000 0x074B0B7D

sleep 100

# =====
# 399MHz - 26MHz input, PD=1,MFI=7, MFN=27, MFD=40
# =====
mww 0x53F80004 0xFF871D50
mww 0x53F80010 0x00271C1B
```

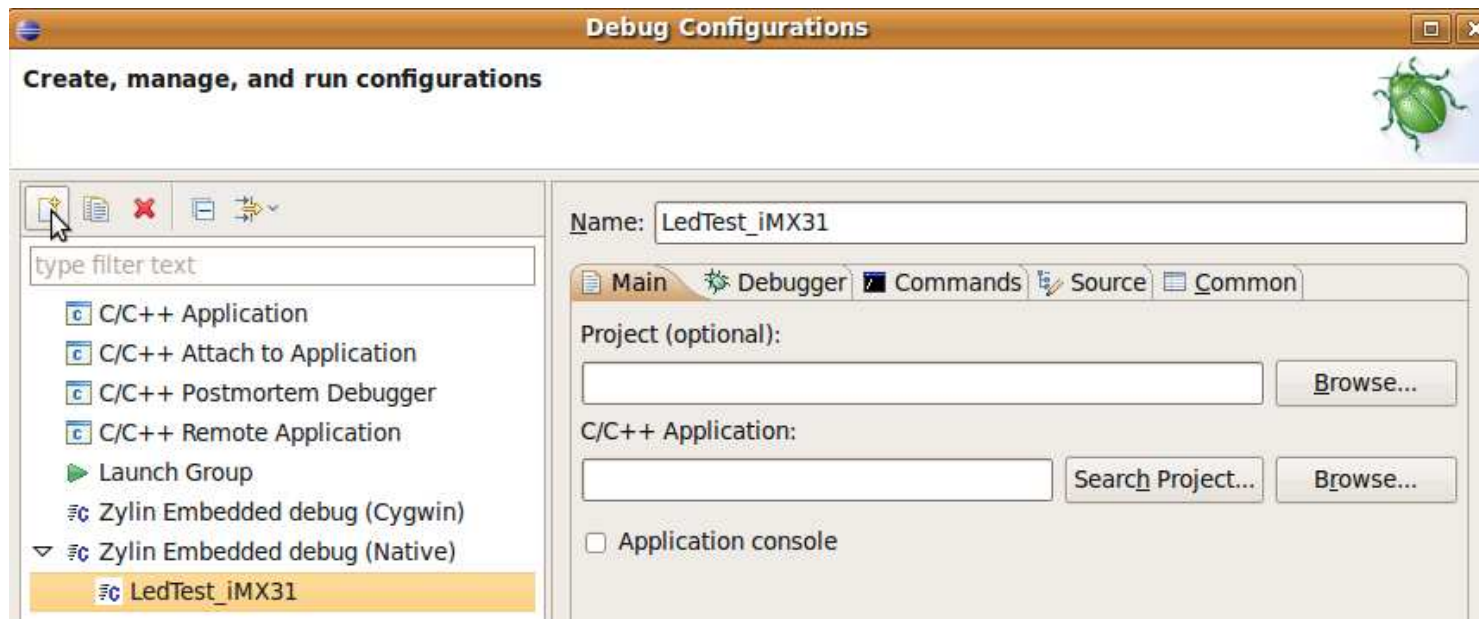
A first debug session – Blinking LEDs

A terminal window titled 'rfrias@rfrias: ~' with a menu bar (File, Edit, View, Terminal, Help). The terminal shows the execution of 'sudo openocd -f interface/signalyzer.cfg -f target/imx31pdk.cfg'. The output includes version information, URLs for bug reports, and device identification details. A red rectangle highlights the JTAG tap detection section.

```
rfrias@rfrias:~$ sudo openocd -f interface/signalyzer.cfg -f target/imx31pdk.cfg
Open On-Chip Debugger 0.3.0-in-development (2009-09-03-00:31) svn:2662
$URL: http://svn.berlios.de/svnroot/repos/openocd/trunk/src/openocd.c $
For bug reports, read http://svn.berlios.de/svnroot/repos/openocd/trunk/BUGS
Can't find target/imx31pdk.cfg
rfrias@rfrias:~$ sudo openocd -f interface/signalyzer.cfg -f board/imx31pdk.cfg
Open On-Chip Debugger 0.3.0-in-development (2009-09-03-00:31) svn:2662
$URL: http://svn.berlios.de/svnroot/repos/openocd/trunk/src/openocd.c $
For bug reports, read http://svn.berlios.de/svnroot/repos/openocd/trunk/BUGS
Info : device: 4 "2232C"
Info : deviceID: 67353760
Info : SerialNumber: 0100542 A
Info : Description: Signalyzer A
Info : clock speed 6000 kHz
Info : JTAG tap: imx31.sjc tap/device found: 0x2b900f0f (mfg: 0x787, part: 0xb90
0, ver: 0x2)
Info : JTAG tap: imx31.cpu tap/device found: 0x07b3601d (mfg: 0x00e, part: 0x7b3
6, ver: 0x0)
Warn : Tap/Device does not have IDCODE
Info : JTAG tap: imx31.smda tap/device found: 0x2190101d (mfg: 0x00e, part: 0x19
01, ver: 0x2)
Info : found ARM1136
```

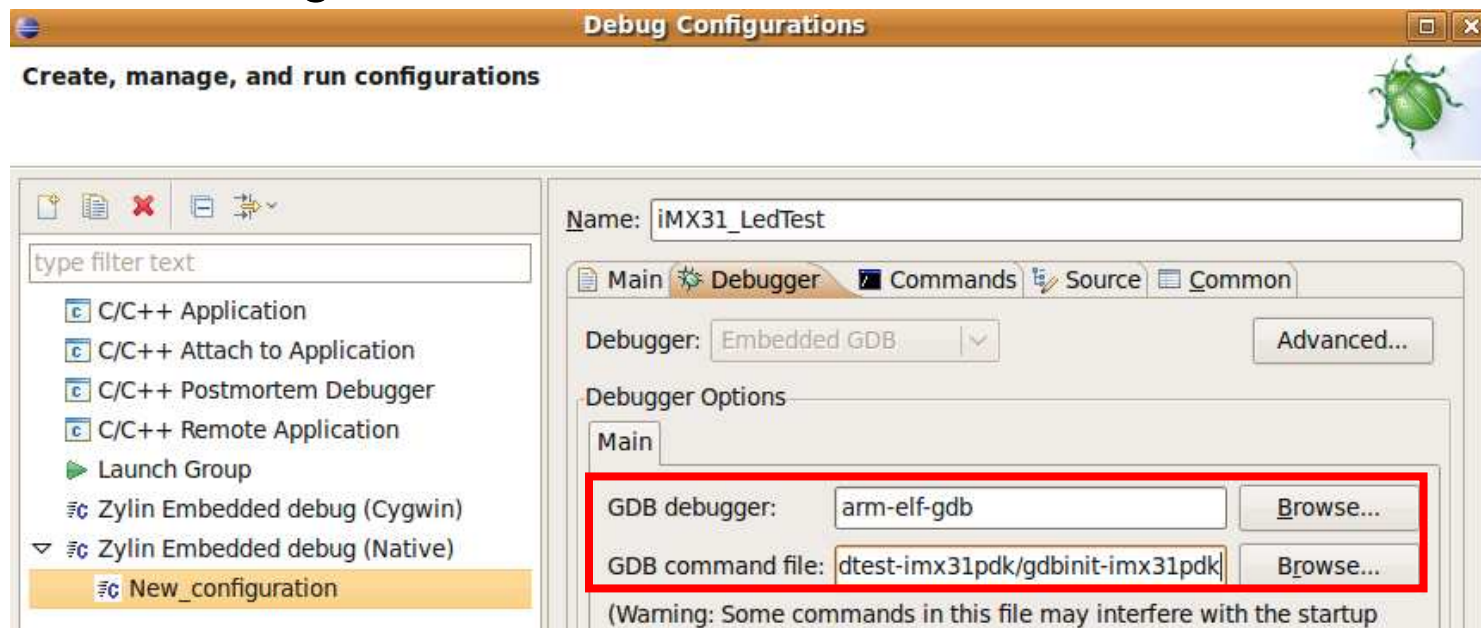
A first debug session – Blinking LEDs

4. Launch Debugger on Eclipse (or command line GDB).
 - Switch to Debug Perspective on Eclipse.
 - Click Debug and Debug Configuration...
 - Select Zylin and create a new debug config
 - Select Project



A first debug session – Blinking LEDs

4. Launch Debugger on Eclipse (or command line GDB).
 - Click the Debugger tab.
 - Change GDB-debugger to arm-elf-gdb
 - Select GDB command file: gdbinit-imx31pdk
 - Click Debug.



A first debug session – Blinking LEDs

- ELF file and GDB init

```
26: 80000100    0 NOTYPE  GLOBAL DEFAULT    1 _mainCRTStartup
27: 80000300    0 NOTYPE  GLOBAL DEFAULT    ABS __bss_end__
28: 80000100    0 NOTYPE  GLOBAL DEFAULT    1 _start
29: 80000180   64 FUNC     GLOBAL DEFAULT    1 main
30: 8000013c   68 FUNC     GLOBAL DEFAULT    1 delay
31: 800001e8   16 FUNC     GLOBAL DEFAULT    1 atexit
```

```
# CONNECT TO TARGET :
target remote 127.0.0.1:3333

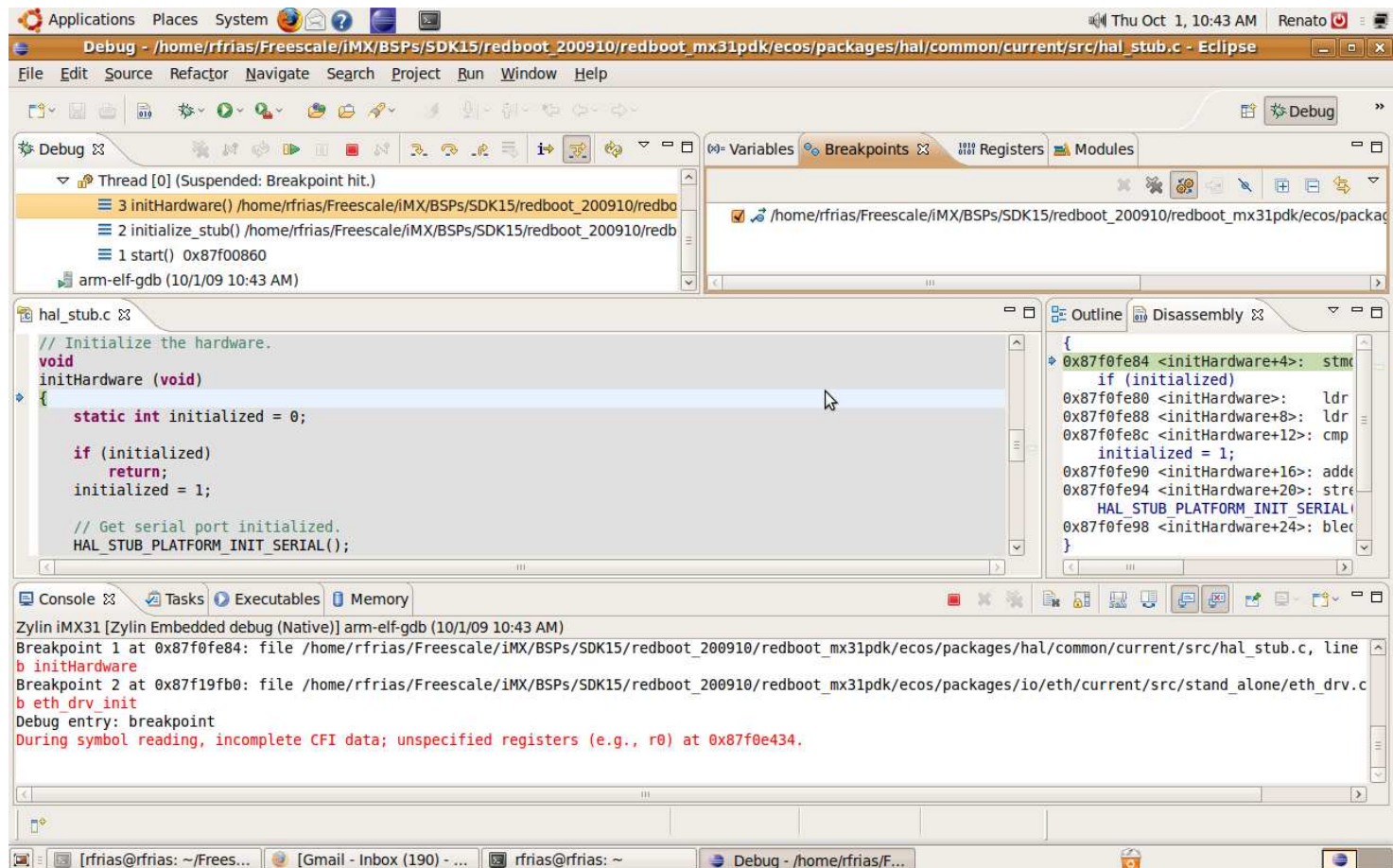
# LOAD IMAGE :
#

# Load the test file
load test.elf

# Load the symbols
symbol-file test.elf
```


A first debug session – Blinking LEDs

5. Debug.



A typical debug session – Redboot

This toolset can be used to debug the bootloader.

Once redboot is running on the DDR it is possible to initialize the NFC and program the NAND.

Once Redboot is running on target board use the commande:

```
Redboot> factive NAND
```

```
Redboot> romupdate
```




Installing the tools

GNU ARM Toolchain 1/2

- Usage:

Cross Compiling tools for ARM.

- Source:

Freescale i.MX BSP



- Install instructions:

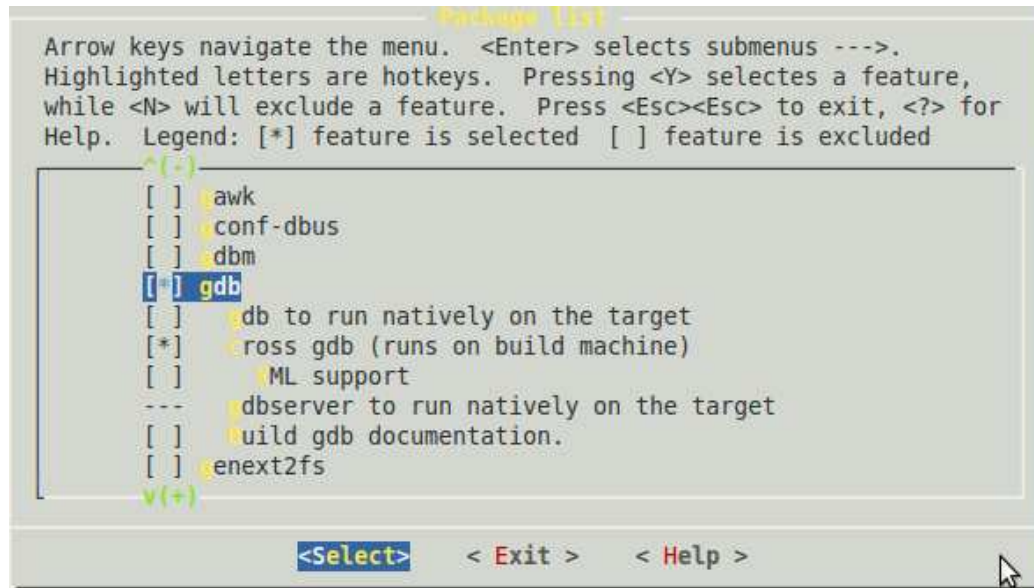
After BSP install the toolchain resides in:

`/opt/freescale/usr/local/gcc-4.1.2-glibc-2.5-nptl-3/arm-none-linux-gnueabi`

GNU ARM Toolchain 1/2

- **Install instructions (cont):**

After BSP installation run “ltib –c” and select GDB on package list section:



The screenshot shows a terminal window with the 'Package List' menu. At the top, instructions state: 'Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> will exclude a feature. Press <Esc><Esc> to exit, <?> for Help. Legend: [*] feature is selected [] feature is excluded'. The menu lists several packages: awk, conf-dbus, dbm, **[*] gdb**, gdb to run natively on the target, [*] cross gdb (runs on build machine), ML support, dbserver to run natively on the target, build gdb documentation, and next2fs. The 'gdb' option is highlighted with a blue bar and a yellow asterisk. At the bottom, there are three buttons: '<Select>', '<Exit>', and '<Help>'. A mouse cursor is visible over the '<Exit>' button.

Copy “gdb” from <ltib install folder>/bin/gdb to /usr/bin/arm-elf-gdb :
\$ sudo cp ltib/bin/gdb /usr/bin/arm-elf-gdb

OpenOCD (Open On-Chip Debugger) – 1/2

- **Usage:**

Provides means to Eclipse/CDT debugger interface with remote target through GDB connection.

- **Source:**

Open Source project initiated by Dominic Rath and maintained by several developers.

<http://openocd.berlios.de/web/>

- **Install instructions:**

Download stable version from:

<http://developer.berlios.de/projects/openocd>

Or get latest release from SVN:

\$ *svn checkout* <http://svn.berlios.de/svnroot/repos/openocd/trunk> *openocd*

- Install instructions (cont.):

After downloading the source from website or SVN, do:

```
$ cd openocd
$ ./bootstrap
$ ./configure --enable-<your probe option>*
$ make
$ sudo make install
```

Detailed install instructions can be found on Readme file on openocd folder.

* On this presentation case, to configure for a FTDI based probe: `$./configure --enable-ft232rl-ft232rl`

Eclipse IDE – 1/3

- **Usage:**

Graphical integrated development environment. Can be used to cross compile on this case will be used to debug.

- **Source:**

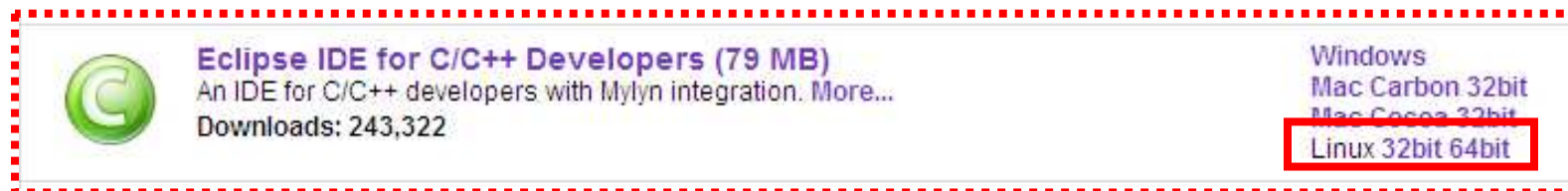
Latest Eclipse release is “Galileo”.

<http://www.eclipse.org/>

- **Install instructions:**

Go to downloads and get C/C++ package (CDT).

<http://www.eclipse.org/downloads/>

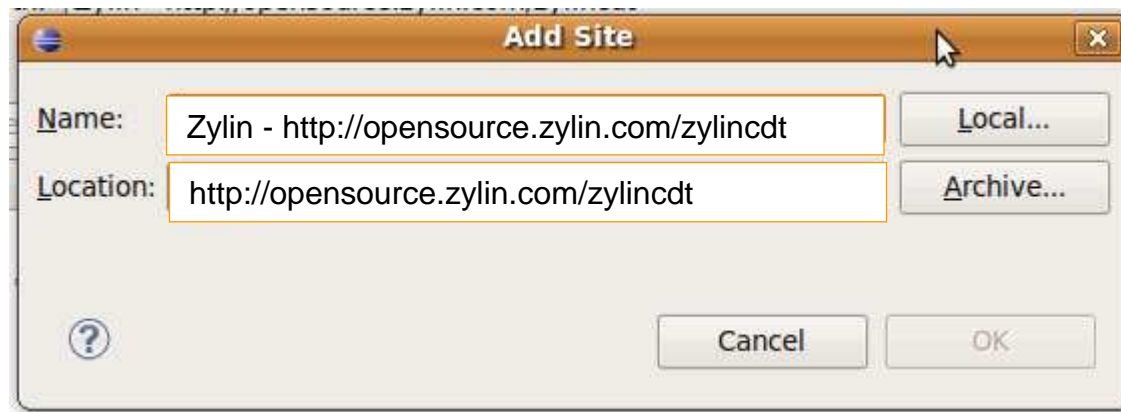


Eclipse IDE – 2/3

- **Install instructions:**

After downloading the Eclipse Package, extract it using “tar” and run it. There are several Eclipse install tutorials available on the net.

Debug features on Eclipse does not allow connecting to the embedded processor directly, thus we need to install Zylind plugin. On Eclipse “Help” Menu item, chose “Software Updates”. Click on add and enter:



Then click “OK”, select Zylin Embedd CDT on the list and install it

JTAG Probe - 1/4

- **Usage:**

Connects USB port on PC with JTAG port on i.MX SoC.

- **Source:**

There are some vendors selling low cost JTAG probes, few examples:

<http://www.amontec.com/jtagkey2.shtml> - **115 EURO**

<http://www.signalalyzer.com/> - **89 USD**

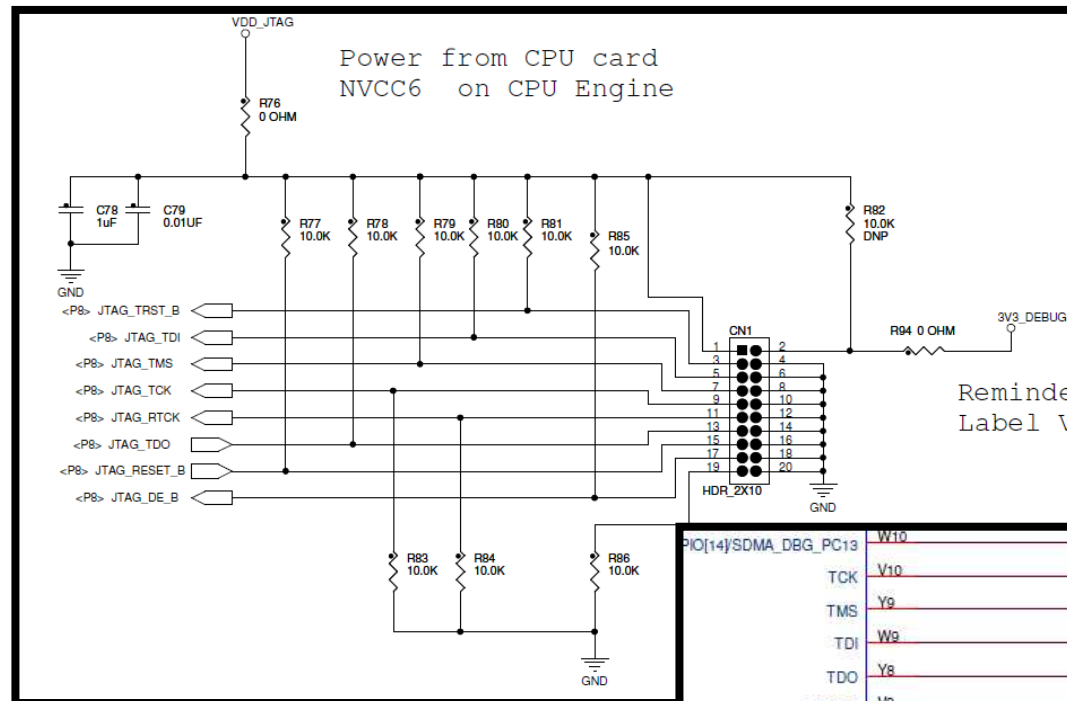
There are some “open hardware” JTAG probe projects based on FTDI USB to Serial chip, FT2322. We can build our own and even embed on PCB.

FT2322 is popular for this application as FTDI provides a JTAG command library for the FT2322, look for: “AN2232C-01_MPSSE_Cmnd.pdf”.

[http://www.ftdichip.com/Documents/AppNotes/AN2232C-01_MPSSE_Cmnd.pdf]

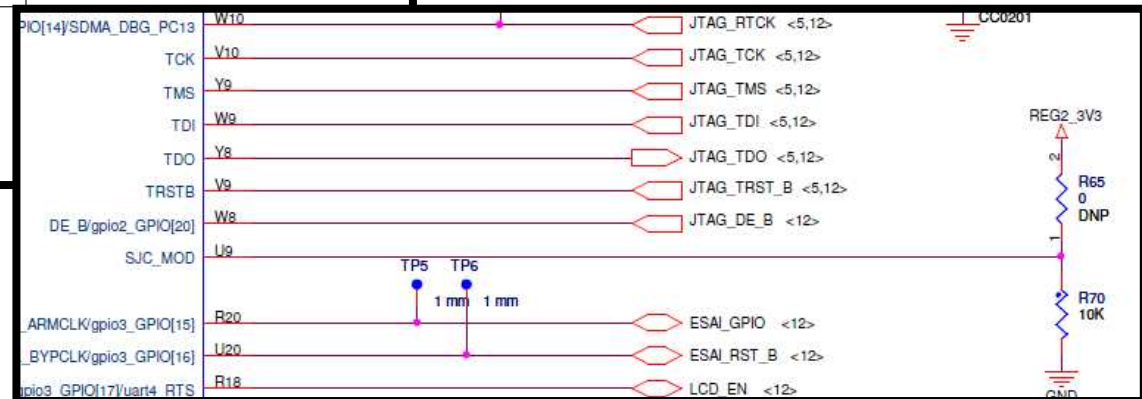
JTAG - 2/4

• JTAG on the PDK:



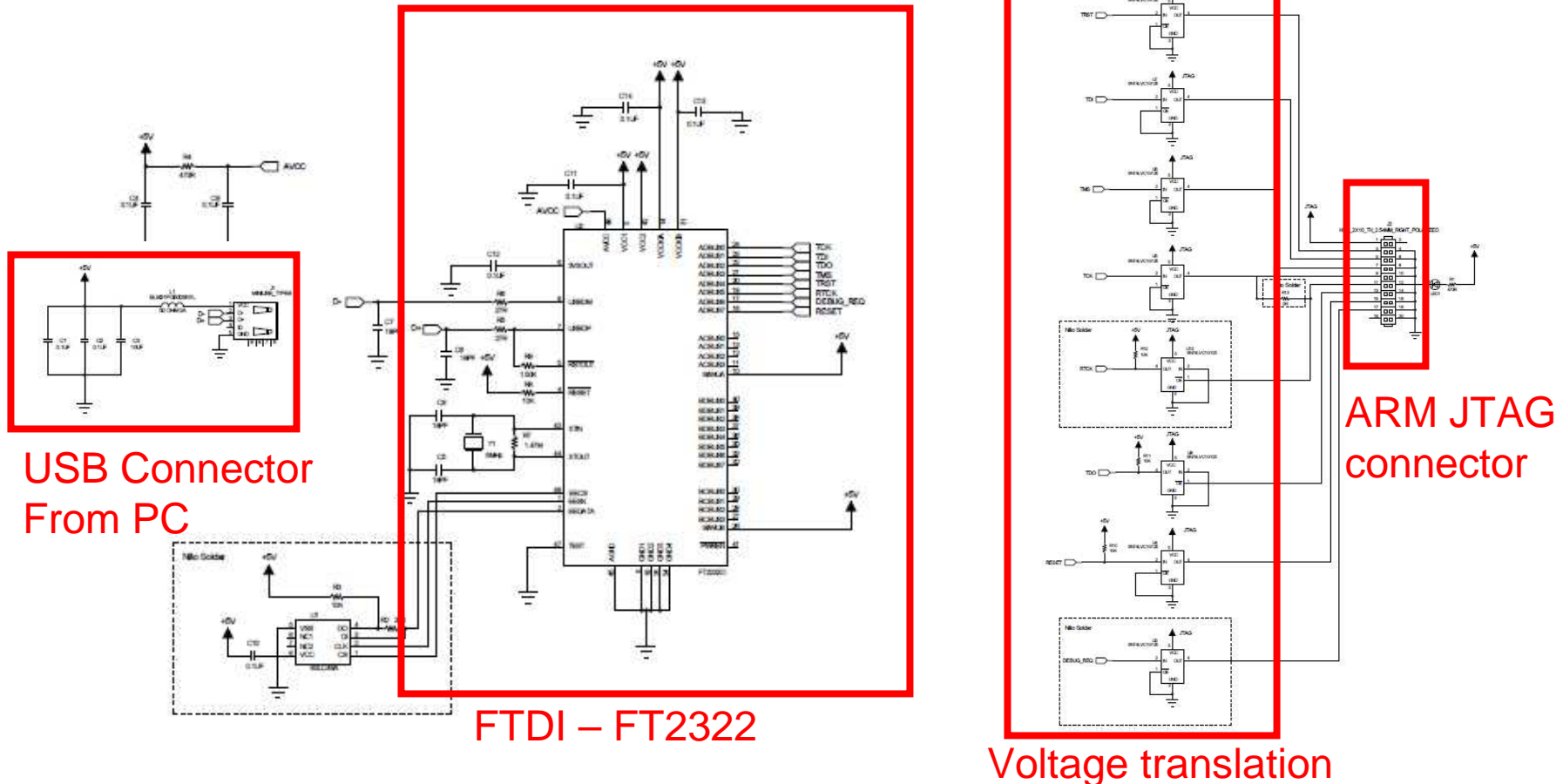
Debug Board

CPU Board



JTAG Probe - 3/4

- A really simple probe:



JTAG Probe - 4/4

- **Install Instructions:**

FTDI drivers are needed on host PC, there are two drivers available, open source and proprietary. The latter is optimized, so let's use it.

Download the drivers:

<http://www.ftdichip.com/Drivers/D2XX.htm>

Current release for Linux was 0.4.16 at the time this document was written.

Linux	FT2232H, FT4232H, FT232R, FT245R, FT2232, FT232B, FT245B, 0.4.16 FT8U232AM, FT8U245AM	2nd December 2008	Instructions in ReadMe file.
-------	---	-------------------	--

Follow instructions from **ReadMe** file.

Or execute the commands listed on:

http://www.imxdev.org/wiki/index.php?title=IMX27_ADS_Board_Installing_OpenOCD_and_GDB

