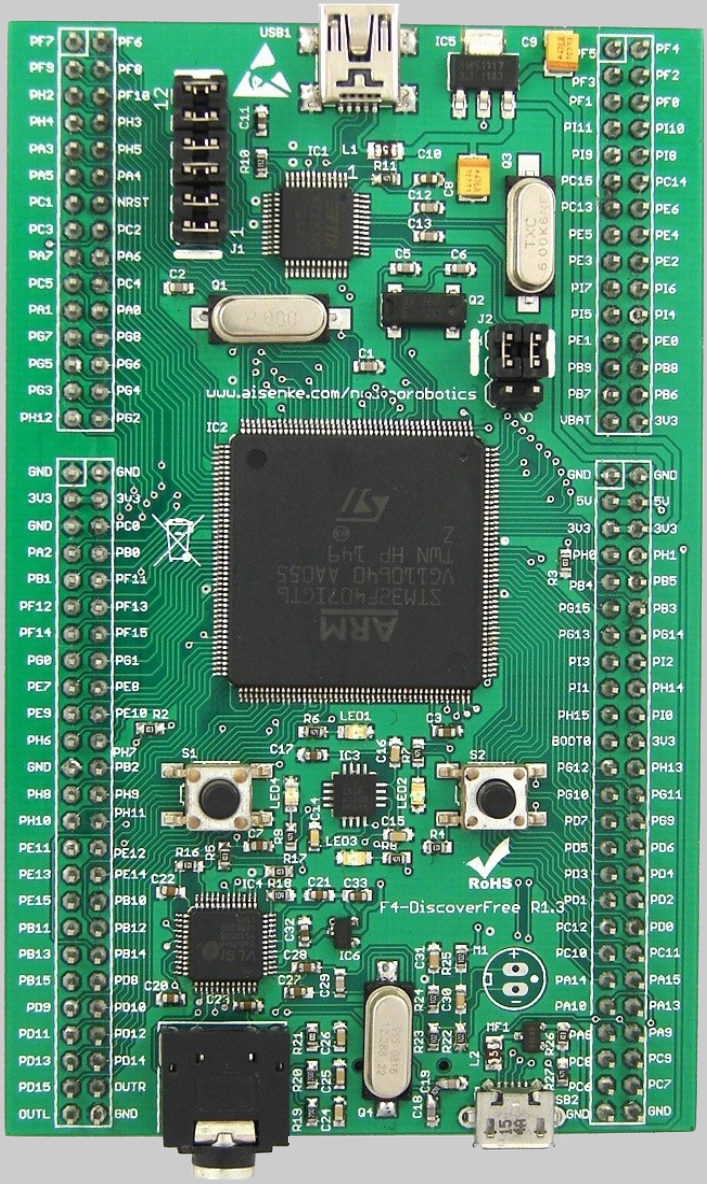
F4-DiscoverFree Kit Series

DATA BRIEF

F4-DiscoverFree Board

Version R1.3

DB-F4-0001



Chongqing Aisenke Electronic Technology Co.,Ltd.

Phone/Fax: +86-2363743515
Email: info@aisenke.com

Data Brief for F4-DiscoverFree Board R1.3

This data brief introduce the Free and Open Source development board F4-DiscoverFree.

Revision 1.3, January 2016

Nodino Robotics is a registered trademark of Aisenke Electronic Technology Co.,Ltd. For more information about this project and F4-DiscoverFree, visit us online at http://www.aisenke.com

Table of Contents

Illustration Index

Index of Tables

1 Introduction

F4-DiscoverFree Board is the only thing you need to discover the most popular STM32F4 32-bit ARM Cortex™-M4 based MCU featuring: all-in-one development board, embedded FT2232 JTAG for debugging and programming, and some external peripherals to get started quickly such as accelerometer, push buttons, LEDs, microSD slot and audio codec.

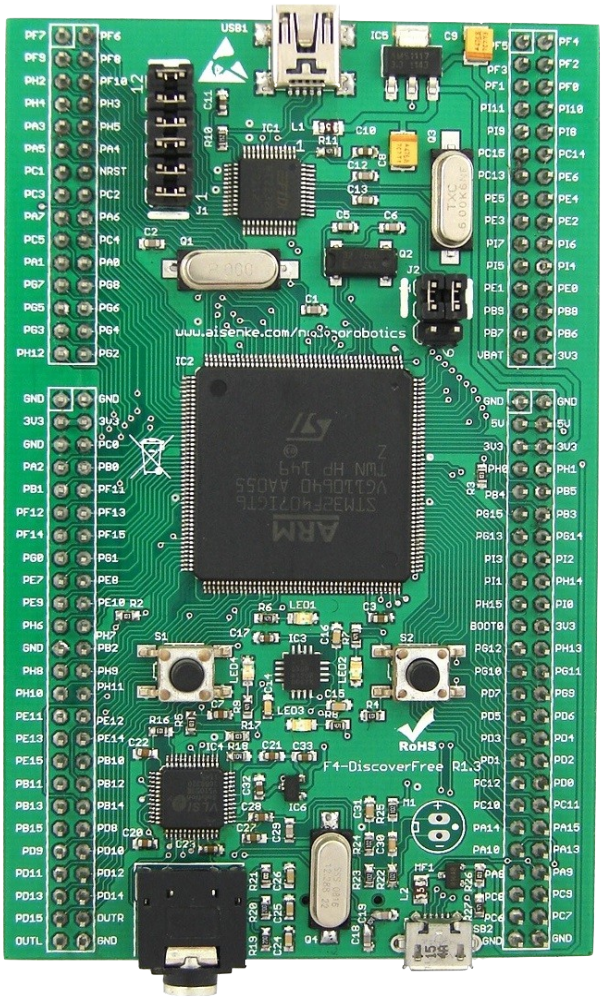


Illustration 1: F4-DiscoverFree board from top view

The core microcontroller is STM32F407IGT6 with 1M of Flash, 192KB of RAM and some built-in peripherals like USART, SPI, I2C, I2S, USB, CAN, SDIO, ADC, DAC, Timer, Ethernet and Camera Interface. This STM32F4 also includes hardware FPU for quicker floating point math calculation.

F4-DiscoverFree board is suitable for beginners to advanced users developing simple to sophisticated electronic projects in no time!

2 Hardware References

Figure 2 and figure 3 briefly explain peripheral blocks and interfaces available on top and bottom layers of F4-DiscoverFree:

- Section 1 is an on-board FT2232 JTAG Debugger and Programmer with two groups of jumper settings.
- Section 2 is the STM32F407IGT6 core microcontroller in LQFP176 package.
- Section 3 contains external peripherals: two push buttons, 3-axis accelerometer, 4 LEDs, 3.5mm stereo audio socket, micro USB AB connector.

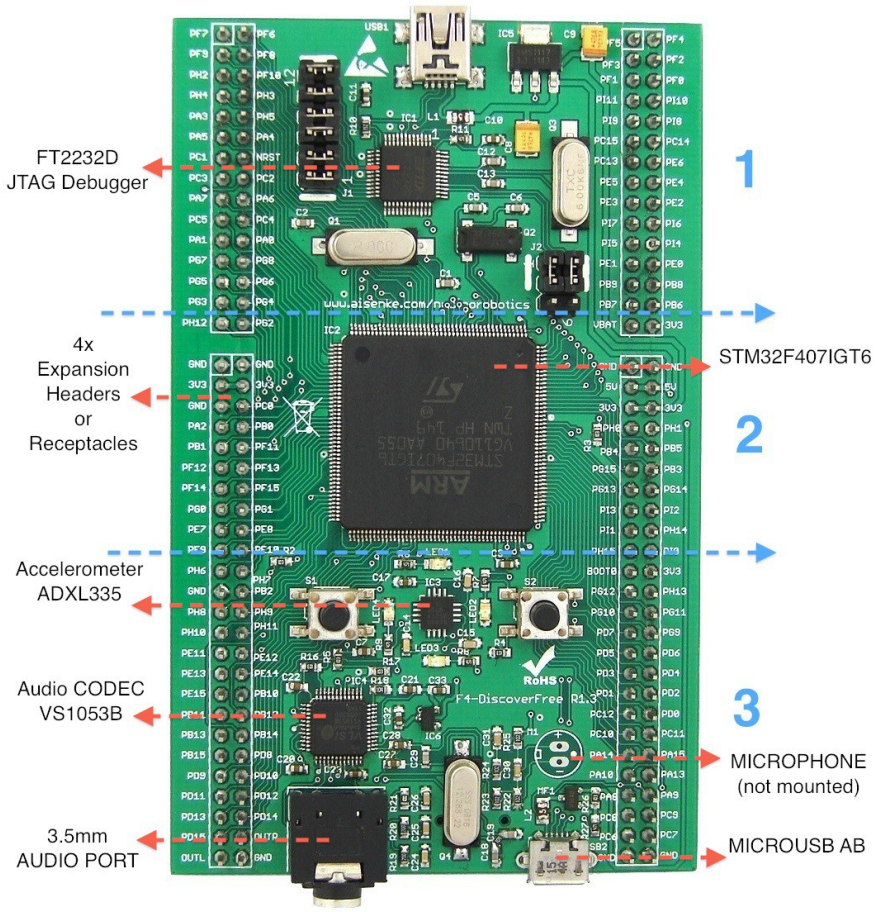


Illustration 2: F4-DiscoverFree top component description

- LED1-LED4 colors are yellow, red, blue and green.
- Push button S1 is a RESET button.
- Push button S2 is a user programmable button.
- ADXL335 is an analog 3-axis accelerometer connected to

ADC inputs of STM32F4.

- VS1053B audio codec supports several file format such as OGG Vorbis, MP3, AAC, WMA, FLAC and MIDI.
- Stereo audio socket can be used with a standard 3.5mm stereo audio jack. The audio output can also be wired from the expansion headers.

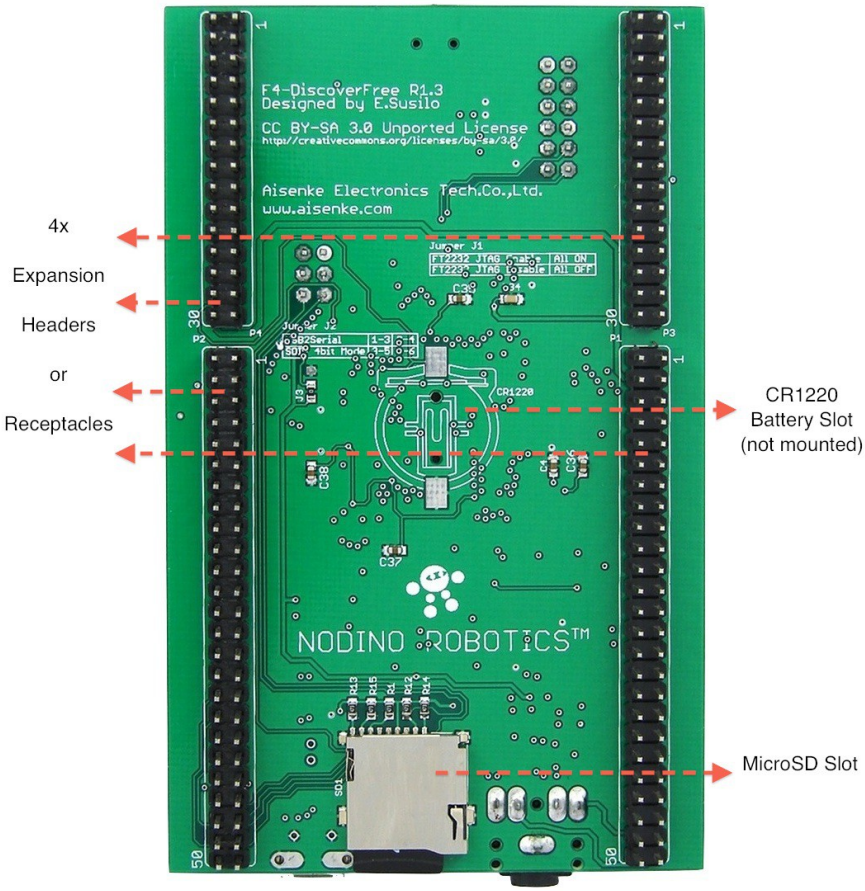


Illustration 3: F4-DiscoverFree bottom component description

- MicroSD slot is of a push-push type. Push to insert and push to eject the card.
- At bottom layer, two lines containing each 2x15 and 2x25 headers are board interconnection to any extension board for users.

3 Getting Started and Requirement

Building your first application on F4-DiscoverFree requires you to setup a development environment on your system. In order to use F4-DiscoverFree with a commercial toolchain, you need to read their respective manual on how to interface to OpenOCD through GDB server. This topic is not within the scope of this manual. However, you may search online for tutorials or contact the vendor for further assistance.

The minimum requirement to develop a standalone application on F4-DiscoverFree are:

- PC or Intel Mac with at least one USB port
- OS supported: Windows, Linux and Mac OS X
- USB type A to mini-B cable
- USB micro-A/B to standard A receptacle (optional)
- MicroSD card reader (optional)

Along with the spirit of the free and open source software, we recommend users to use Eclipse CDT as the Integrated Development Environment, Launchpad GCC for ARM Embedded as the toolchain, and OpenOCD for debugging and programming. There is absolutely no requirement to disclose any proprietary information developed from scratch using any open source tools. Detail installation are provided for Windows, Linux and Mac OS X.

3.1 Getting Started on Windows

3.1.1 Package Installer

For impatient users, you can visit our website and download a package installer and run it. This package installer contains:

- Software and driver
- GCC ARM embedded toolchain
- OpenOCD
- Eclipse IDE fully configured for programming and debugging
- Project template and examples

You have also an option to manually download and install all packages by yourself. Refer to the following subsection for Manual Installation.

3.1.2 Manual Installation

Quick installation instructions on Windows:

- Install Java for Windows

- Install MSYS and Git for Windows
- Install FTDI driver and libusb-win32-devel-filter
- Install GitHub for Windows (optional)
- Install GCC ARM Embedded Toolchain
- Install OpenOCD Debugger

Please read their respective guide/manual for detail and visit our website for download links.

3.1.3 Manual Installation Notes

Java:

Java Run Time Environment installation is required for Eclipse IDE. In case of JRE is already installed, you may skip this step and continue.

MSYS and Git:

Since most of free and open source software developments are available for Unix-like environment, Windows users need to install MSYS in order to provide a similar one. MSYS is not Unix on Windows. It only provides utilities such as make, rm and grep to allow building of applications and programs which depend on UNIX tools to be present. Git for Windows installation is needed as a supplement to git command line.

FTDI and libusb-win32-devel-filter:

It is recommended to get the latest driver of FTDI chip directly from their website. After both FTDI driver and libusb-win32-devel-filter are set, connect the USB mini A of F4-DiscoverFree board to computer. Wait until the system is configured properly. Go to Programs - LibUSB-Win32 and click Filter Wizard. Choose to install a device filter. Within device listing, users should see something like this:

vid:0403 pid:6010 USB Serial Converter A

vid:0403 pid:6010 USB Serial Converter B

Select only USB Serial Converter A and leave port B as it is. Click install and done!

Explanation: By default, FTDI driver assigns FT2232D chip as dual serial port. The libusb-win32-devel-filter role is to make port A "driverless" so that OpenOCD can take control and use it as JTAG Debugger. Without filtering it, OpenOCD will not be able to claim the port.

GCC ARM Embedded:

Installing GCC ARM Embedded toolchain is very straightforward. The binary is available to download and packed as an installer. The default install path should work just fine, or can be customized. For example, to simplify pathname, users may install it at C:\gcc-arm-none-eabi.

OpenOCD:

Instead of building from source, Freddie Chopin has the binary already built for Windows XP, Vista, 7 and 8. Extract the downloaded file and check out the content of bin folder. Rename openocd-x.y.z.exe into openocd.exe. Copy all of files within this folder to where the toolchain binaries are installed. For example, if the GCC ARM Embedded binaries were installed in C:\gcc-arm-none-eabi\bin, then copy all of OpenOCD binaries in here as well.

The next thing to do is copying the scripts folder to toolchain's share folder. For example, if the toolchain was installed in C:\gcc-arm-none-eabi then copy the scripts folder into C:\gcc-arm-none-eabi\share\openocd. The openocd folder might not exist, but can be easily created.

A configuration file for FT2232D based JTAG is required before we can program and debug an application using OpenOCD. This configuration file is not included by default within OpenOCD installation. Download it from our website. The configuration file name is ft2232d.cfg, and put under C:\gcc-arm-none-eabi\share\openocd\scripts\interface

3.2 Getting Started on Linux

3.2.1 Overview

Users are expected to know how to edit some text files and typing commands through terminal. Knowledge in scripting language may help although is not necessary. Steps below are tested to work on Ubuntu Linux and should also work on other linux distribution.

Quick installation instructions on Linux:

- Install package dependencies
- Install Java
- Install/build GCC ARM Embedded Toolchain
- Build OpenOCD Debugger
- Create udev rules for users to access USB port

Please read their respective guide/manual for detail. Steps provided the following are merely the hack, enhancement and manual installation of software/driver not available by default from chip vendors.

3.2.2 Package Dependencies

Use Synaptic Package Manager to install package dependencies or use command line:

```
sudo apt-get install build-essentials \
gcc-mingw32 libusb libftdi libtool autoconf \
automake git picocom
```

A brief explanation of what they are: build-essentials is a native Linux toolchain; gcc-mingw32 is a cross-compile toolchain for

Windows; autoconf and automake are needed for building GCC toolchain; libusb, libftdi and libtool are dependencies to build OpenOCD; git is a versioning system; picocom is a simple debugging tool over serial port.

3.2.3 Java Installation

Use Synaptic Package Manager to install Java or use command line:

```
sudo apt-get install default-jre
```

3.2.4 GCC ARM Embedded Toolchains

GCC for ARM Embedded binaries are available for Linux, Windows and Mac users. The ready-to-use binaries for Linux can be downloaded from LaunchPad. Users are encourage to download this binaries since they are now supported and maintained regularly by Launchpad. Known version to work and tested with F4-DiscoverFree is gcc-arm-none-eabi-4_7-2014q2-20140408-linux.tar.bz2

Instructions provided below are for building the toolchain from source using a build script. Open terminal application and obtain the build script using git:

```
cd ~

git clone http://github.com/ekawahyu/gcc-arm-embedded-macosx

cd gcc-arm-embedded-macosx
```

Leave the terminal application open and download the source code from Launchpad's repository for the bare-metal system without OS. Download gcc-arm-none-eabi-4_6-2012q2-20120614-src.tar.bz2 and put them in gcc-arm-embedded-macosx folder. Execute the build script under terminal application:

```
./build-linux.sh
```

Get some coffee or do something else since this might take a little longer. Binaries will be available under install-linux folder.

Move the binaries and rename the folder name by typing:

```
mv install-linux ../gcc-arm-none-eabi
```

Test the toolchains by typing:

```
cd ~/gcc-arm-none-eabi/bin

./arm-none-eabi-gcc -v
```

Output should show something like this:

```
Using built-in specs.
COLLECT_GCC=./arm-none-eabi-gcc
.
.
.
Thread model: single
gcc version 4.6.2 20120613 (release)
[ARM/embedded-4_6-branch revision 188521] (GNU
Tools for ARM Embedded Processors)
```

Done building the toolchains!

3.2.5 OpenOCD Debugger

Users need to download OpenOCD Debugger source code and extract it under home folder. OpenOCD binary and shares will be installed at the same location of the toolchain. For example, if the username is nodino, then /home/nodino/gcc-arm-none-eabi is where we have installed the toolchains. As of this writing, the script is tested to work on OpenOCD-0.8.0. Open terminal and start building OpenOCD by typing commands below:

```
./configure --prefix=/home/nodino/gcc-arm-none-
eabi --enable-ftdi --enable-stlink --enable-jlink
make
```

After the build is done, install it by typing:

```
make install
```

A configuration file for FT2232D based JTAG is required before we can program and debug an application using OpenOCD. This configuration file is not included by default within OpenOCD installation. Download it from our server, the ft2232d.cfg configuration file, and put under ~/gcc-arm-none-eabi/share/openocd/scripts/interface/ftdi

Test the OpenOCD Debugger by typing:

```
cd ~/gcc-arm-none-eabi/bin
./openocd -v
```

Output should show something like this:

```
Open On-Chip Debugger 0.8.0 (2014-06-28-23:36)
Licensed under GNU GPL v2
For bug reports, read
http://openocd.sourceforge.net/doc/doxygen/bugs.h
tml
```

Done building the debugger!

3.2.6 Set Permission for Users

In order to give users permission to access USB port, udev rules is needed. Open terminal application and create it using nano text editor:

```
sudo nano /etc/udev/rules.d/45-ftdi2232-
libftdi.rules
```

Copy and paste the line below:

```
ATTR{idProduct}=="6010", ATTR{idVendor}=="0403",
MODE="666", GROUP="plugdev"
```

Save the file and exit from nano text editor. Users should now be able to gain access to USB port.

3.3 Getting Started on Mac OS X

3.3.1 Overview

Users are expected to know how to edit some text files and typing commands through terminal. Knowledge in scripting language may help although is not necessary.

Quick installation instructions on Mac OS X:

- Install Java for Mac OS X
- Install Xcode and please read notes
- Install Macports and some ports required
- Install FTDI driver and the hack
- Install Git-OSX-Installer
- Install GitHub for Mac OS X (optional)
- Build GCC ARM Embedded Toolchain
- Build OpenOCD Debugger

Please read their respective guide/manual for detail. Steps provided the following are merely the hack, enhancement and manual installation of software/driver not available by default from chip vendors.

3.3.2 Xcode Installation Notes

Xcode is an Integrated Development Environment (IDE) containing a suite of software development tools developed by Apple for developing software for OS X and iOS. Xcode full installation is required for Snow Leopard users, but not necessary for Lion, Mountain Lion and Mavericks users. Apple has separated the software development tools from Xcode package installer and is

available online to download, known as Xcode command line tools.

For Snow Leopard users, Xcode 3.2 is included within the installation DVD. For Lion, Mountain Lion and Mavericks, the command line tools can be downloaded from Apple Developer Download.

Execute this line after installing Xcode or command line tools:

```
xcode-select --install
```

Execute this line after installing Xcode:

```
sudo xcodebuild -license
```

3.3.3 Macports Ports

Users need to download and install Macports prior to adding some packages required. After the installation is done, open a terminal and execute:

```
sudo port -v selfupdate
```

Install some packages and their dependencies:

```
sudo port install autoconf automake libusb
libftdi1 libtool picocom
```

A brief explanation of what they are: autoconf and automake ports are needed for building GCC toolchain; libusb, libftdi and libtool are dependencies to build OpenOCD. Picocom is a simple debugging tool over serial port.

3.3.4 FTDI Driver Hack

Users need to download and install the FTDI Virtual COM Port driver from FTDI website before applying this hack. The purpose of applying it is to get the FT2232 JTAG and serial port to work side by side under Mac OS X.

Edit the /System/Library/Extensions/FTDIUSBSerialDriver.kext/ Contents/Info.plist file and remove the following lines:

```
<key>FT2232C_A</key>
<dict>
<key>CFBundleIdentifier</key>
<string>com.FTDI.driver.FTDIUSBSerialDriver</stri
ng>
<key>IOClass</key>
<string>FTDIUSBSerialDriver</string>
<key>IOProviderClass</key>
<string>IOUSBInterface</string>
<key>bConfigurationValue</key>
<integer>1</integer>
<key>bInterfaceNumber</key>
<integer>0</integer>
<key>idProduct</key>
<integer>24592</integer>
<key>idVendor</key>
<integer>1027</integer>
</dict>
```

By removing those lines, the FT2232 serial driver on port A will be free for libusb to take over, while port B will still be using FTDI USB Serial driver.

Reload the driver by executing the following lines under terminal application:

```
sudo kextunload /System/Library/Extensions/
FTDIUSBSerialDriver.kext
sudo kextload /System/Library/Extensions/
FTDIUSBSerialDriver.kext
sudo touch -m /System/Library/Extensions
```

3.3.5 AppleUSBFTDI Driver Hack for OS X 10.9 and 10.10

Starting from Mac OS X 10.9 (Mavericks), Apple includes an FTDI Virtual COM Port kernel extension. The purpose of applying this hack is to get the FT2232 JTAG and serial port to work side by side under Mac OS X.

The quickest workaround, on OS X 10.9, to disable the built-in FT2232 serial driver from Apple is:

```
sudo mv
/System/Library/Extensions/IOUSBFamily.kext/Conte
nts/PlugIns/AppleUSBFTDI.kext
/System/Library/Extensions/IOUSBFamily.kext/Conte
nts/PlugIns/AppleUSBFTDI.disabled
```

The quickest workaround, on OS X 10.10, to disable the built-in FT2232 serial driver from Apple is:

```
sudo mv
/System/Library/Extensions/AppleUSBFTDI.kext
/System/Library/Extensions/AppleUSBFTDI.disabled

sudo nvram boot-args="kext-dev-mode=1"
```

This way the kernel extensions is disabled at booting time. The nvram command is necessary to put OS X 10.10 into kext development mode. Otherwise, no serial port detected.

3.3.6 AppleUSBFTDI Driver Hack for OS X 10.11

Apple has enabled a new default security oriented featured called System Integrity Protection (SIP), often called rootless, in OS X 10.11 onward. The rootless feature is aimed at preventing Mac OS X compromise by malicious code, whether intentionally or accidentally, and essentially what SIP does is lock down specific system level locations in the file system while simultaneously preventing certain processes from attaching to system-level processes. SIP locks down the following system level directories in OS X: /System, /sbin, /usr (with exception of /usr/local.

No root access is allowed to manipulate SIP protected directories. In order to disable SIP temporarily to make some changes, restart OS X, press and hold Cmd+R after you hear Apple chime and then release it as soon as Apple logo appears.

Open terminal application and type:

```
Csrutil enable –without kext
```

It will enable SIP but disable kext signing. Restart OS X one more time, reenter recovery mode, and type this on terminal application:

```
mv
/System/Library/Extensions/AppleUSBFTDI.kext
/System/Library/Extensions/AppleUSBFTDI.disabled
```

This will exclude AppleUSBFTDI.kext from being uploaded during OS X boot. You can restart your OS X and install the latest FTDI driver. Please read FTDI Driver Hack for further information and DO THE HACK during recovery mode or SIP disabled.

3.3.7 GCC ARM Embedded Toolchains

GCC for ARM Embedded binaries are available for Linux, Windows and Mac users. The ready-to-use binaries for Mac can be downloaded from LaunchPad. Users are encourage to download this binaries since they are now supported and maintained regularly by Launchpad. Known version to work and tested with F4-DiscoverFree is gcc-arm-none-eabi-4_7-2014q2-20140408-

mac.tar.bz2

Building the toolchains from source is straightforward using a build script. Open terminal application and obtain the build script using git:

```
cd ~

git clone http://github.com/ekawahyu/gcc-arm-embedded-macosx

cd gcc-arm-embedded-macosx
```

Leave the terminal application open and download the source code from Launchpad's repository for the bare-metal system without OS. Download gcc-arm-none-eabi-4_6-2012q2-20120614-src.tar.bz2 and put them in gcc-arm-embedded-macosx folder. Execute the build script under terminal application:

```
./build-macos.sh
```

Get some coffee or do something else since this might take a little longer. Binaries will be available under install-macosx folder.

Move the binaries and rename the folder name by typing:

```
mv install-macosx ../gcc-arm-none-eabi
```

Test the toolchains by typing:

```
cd ~/gcc-arm-none-eabi/bin

./arm-none-eabi-gcc -v
```

Output should show something like this:

```
Using built-in specs.

COLLECT_GCC=./arm-none-eabi-gcc

.

.

.

Thread model: single

gcc version 4.6.2 20120613 (release)
[ARM/embedded-4_6-branch revision 188521] (GNU
Tools for ARM Embedded Processors)
```

Done building the toolchains!

3.3.8 OpenOCD Debugger

Users need to download OpenOCD Debugger source code and extract it under home folder. OpenOCD binary and shares will be installed at the same location of the toolchain. For example, if the username is nodino, then /Users/nodino/gcc-arm-none-eabi is where we have installed the toolchains. As of this writing, the script is tested to work on OpenOCD-0.8.0. Open terminal and

start building OpenOCD by typing commands below:

```
LDFLAGS="-L/opt/local/lib" CPPFLAGS="-
I/opt/local/include" \
./configure --prefix=/Users/nodino/gcc-arm-none-
eabi \
--enable-ftdi --enable-stlink --enable-jlink
make
```

After the build is done, install it by typing:

```
make install
```

A configuration file for FT2232D based JTAG is required before we can program and debug an application using OpenOCD. This configuration file is not included by default within OpenOCD installation. Download it from our server, the ft2232d.cfg configuration file, and put under ~/gcc-arm-none-eabi/share/openocd/scripts/interface/ftdi

Test the OpenOCD Debugger by typing:

```
cd ~/gcc-arm-none-eabi/bin
./openocd -v
```

Output should show something like this:

```
Open On-Chip Debugger 0.8.0 (2014-06-28-23:36)
Licensed under GNU GPL v2
For bug reports, read
http://openocd.sourceforge.net/doc/doxygen/
bugs.html
```

Done building the debugger!

4 Schematic

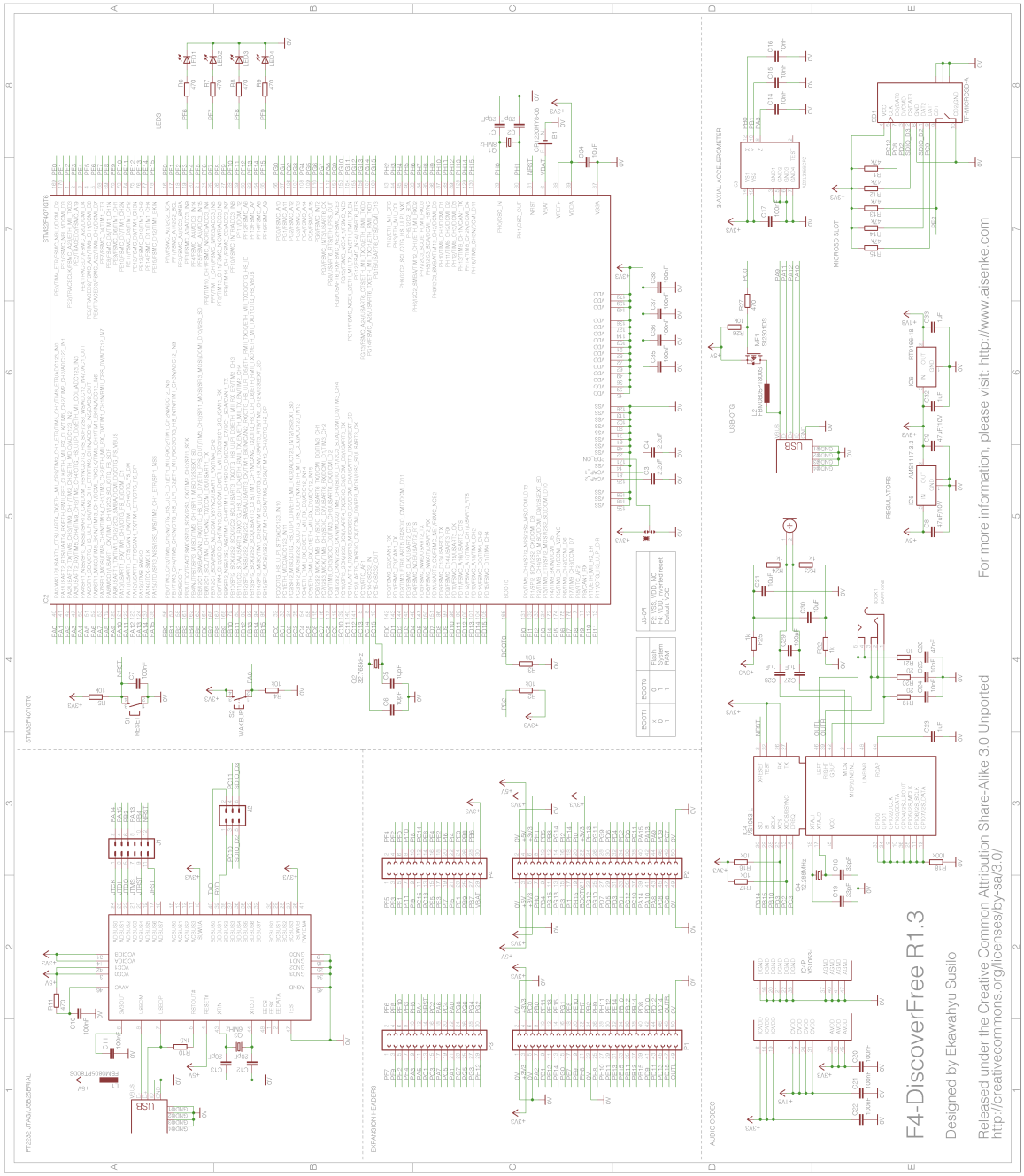


Illustration 4: F4-DiscoverFree Schematic

5 Revision History

Date	Revision	Changes
17 Feb 2013	1	Initial document release
25 Oct 2013	1.1	Added OS X Mavericks Installation
29 June 2014	1.2	Updated OpenOCD to version 0.8.0
12 January 2016	1.3	Added OS X Yosemite and El Capitan Installation

Table 1: Revision history