Detailed Guide

This guides covers two areas, flashing a Bioband and how to set up the cross compile software on a new/clean Linux PC (best described as a memory guide to the steps involved).

Flashing a Bioband

The Bioband is programmed using the JTAG interface. This requires a device be used to talk to the JTAG interface from the PC.

For this project the Amontec JTAGkey has been used (www.amontec.com), although other devices should provide the same function. The use of this device was a quick solution when combined with the open source openocd program for controlling the JTAG process.

Openocd

In a terminal, change to the bioband/band/gcc directory, i.e

```
cd /home/mrc/bioband2/band/gcc
```

Then run openocd

```
openocd
```

Depending on the version installed you should get output similar to the following (although the version of openocd is now likely to be 0.3.1)

```
Open On-Chip Debugger 0.2.0 (2009-12-11-10:48) svn:2826
$URL: http://svn.berlios.de/svnroot/repos/openocd/tags/openocd-
0.2.0/src/openocd.c $
For bug reports, read http://svn.berlios.de/svnroot/repos/openocd/trunk/BUGS
500 kHz
jtag_nsrst_delay: 100
jtag_ntrst_delay: 100
Info : JTAG tap: toolchain.cpu tap/device found: 0x3ba00477 (mfg: 0x23b, part:
0xba00, ver: 0x3)
Info : JTAG Tap/device matched
Info : JTAG tap: toolchain.bs tap/device found: 0x16410041 (mfg: 0x020, part:
0x6410, ver: 0x1)
Info : JTAG Tap/device matched
```

Hopefully the output should then remain static.

If there are then intermittant lines on the screen saying something similar to ..

```
Warn : Timeout (1000ms) waiting for ACK = OK/FAULT in SWJDP transaction
```

this means it has detected the device but the Bioband is asleep.

Press the Ctrl and C keys together to break out of the openocd program. Then reawaken the band by either unplugging/replugging it into the USB cable or, if using the breakout board, by pressing the reset button. Then restart openocd.

Flashing the band

With openocd running in a terminal

Open another terminal

Change to the same bioband/band/gcc directory as openocd is running from, compile the code by executing the following

```
make clean
make
```

You should end up with some output similar to the following, only the last few lines shown for brevity

```
<snip>
arm-none-eabi-gcc -I./ -I../inc
-I../../../Libraries/toolchainF10x_StdPeriph_Driver/inc
-I../../../Libraries/CMSIS/Core/CM3 -I../../../Libraries/toolchain_USB-FS-
Device_Driver/inc -c -fno-common -O2 -g -mcpu=cortex-m3 -mthumb
-DtoolchainF10X_LD -DUSE_STDPERIPH_DRIVER -DUSE_toolchain10B_EVAL
-DHSE_VALUE=16000000 -DMRC_CWA ../src/main.c
arm-none-eabi-ld -v -Tlinker.cmd -nostartfiles -o main.out  toolchainf10x_rcc.o
toolchainf10x_gpio.o toolchainf10x_spi.o toolchainf10x_rtc.o toolchainf10x_bkp.o
toolchainf10x_pwr.o toolchainf10x_exti.o toolchainf10x_adc.o
toolchainf10x_usart.o toolchainf10x_tim.o misc.o system_toolchainf10x.o
core_cm3.o startup_toolchainf10x_ld.o toolchainf10x_it.o usb_core.o usb_init.o
usb_int.o usb_mem.o usb_regs.o usb_sil.o hw_config.o usb_desc.o usb_endp.o
usb_istr.o usb_prop.o usb_pwr.o nand_cwa.o accel.o tempsensor.o debug.o shared.o
main.o
GNU ld (GNU Binutils) 2.19
...copying
arm-none-eabi-objcopy -Obinary main.out main.bin
arm-none-eabi-objdump -S main.out > main.list
```

The important part is the main.bin which is transferred to/flashed onto the connected Bioband by executing ..

```
make flash
```

If there is an error finding the script do_flash.pl, make sure that the bioband/scripts directory is on your $PATH by typing `env` into the terminal.

If the scripts directory is missing, type

```
export PATH=/home/mrc/bioband2/scripts:$PATH
```

substituting your script directory path instead of the one given

Upon the make flash, the openocd terminal should give additional output similar to the following

```
Info : accepting 'telnet' connection from 0
Info : JTAG tap: toolchain.cpu tap/device found: 0x3ba00477 (mfg: 0x23b, part:
0xba00, ver: 0x3)
Info : JTAG Tap/device matched
Info : JTAG tap: toolchain.bs tap/device found: 0x16410041 (mfg: 0x020, part:
0x6410, ver: 0x1)
Info : JTAG Tap/device matched
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08007dac
Info : device id = 0x20016410
Info : flash size = 32kbytes
flash 'toolchainx' found at 0x08000000
toolchainx mass erase complete
Warn : not enough working area available(requested 16384, free 16336)
wrote  33572 byte from file
/data/projects/mrc/svn/mrc/software/trunk/dev_board/BioBand2.1/Band/gcc/main.bin
to flash bank 0 at offset 0x00000000 in 2.291996s (14.304194 kb/s)
Info : JTAG tap: toolchain.cpu tap/device found: 0x3ba00477 (mfg: 0x23b, part:
0xba00, ver: 0x3)
Info : JTAG Tap/device matched
Info : JTAG tap: toolchain.bs tap/device found: 0x16410041 (mfg: 0x020, part:
0x6410, ver: 0x1)
Info : JTAG Tap/device matched
Info : dropped 'telnet' connection - error -400
```

Don't worry about the '`not enough working area available`' message

One very important note - after the 'telnet' connection message has been displayed, press the Ctrl and C buttons whilst the focus is on the terminal with openocd running

This step is needed since there is an issue if the bioband goes to sleep (after 5 minutes of inactivity, in order to save battery) whilst openocd is running, the device can become unusable until the battery goes flat and is recharged.

Resetting the Bioband

When flashing a Bioband, it is useful to reset the bkp domain on the device and set the band id.

To do this use the mrc program after flashing

```
./mrc -nobkp
```

You will then need to reset the band (reset button on the breakout board)

Then if flashing for the first time, you will need to assign an identity to the band

```
./mrc -id black1
```

In this example, assigning black1. A short collection is required to save the id into flash.

It is preferable for these ids to be unique, as they are used in part as the USB serial number

This step can also be carried out using the demo program.

Building and setting up the code

Openocd

```
sudo apt-get install libnet-telnet-perl openocd
```

Do a test openocd invocation with a band connected

If the dongle is not found or there are errors try installing libftdi, i.e `sudo apt-get install libftdi1`

Also ensure

```
ft2232_device_desc "Amontec JTAGkey A"
```

or

```
ft2232_device_desc "Amontec JTAGkey-2"
```

is specified in openocd.cfg, in the bioband/band/gcc directory.

Select the correct description depending on the type of Amontec key being used (refer to sticker on the device)

Building the PC side code

On Linux:

Change directory to the Linux directory for the particular PC side code, i.e PC/MRC/Linux or PC/Demo/Linux.

Then type

```
make
```

In order to talk to the Bioband using Demo or Mrc on certain flavours of Linux without being superuser you need to alter the following udev file 50-udev-default.rules in /lib/udev/rules.d

Search for the usb_device line and change the MODE from "0664" to "0666" as shown below

```
SUBSYSTEM=="usb", ENV{DEVTYPE}=="usb_device", MODE="0666"
```

On Windows:

You need to ensure that a copy of Windows Visual Studio is installed, a free copy of Visual Studio 2008 Express is available if you search on the Microsoft website

Double click on MRC.vcproj or Demo.vcproj files in PC/MRC/Windows or PC/Demo/Windows

This will open Visual Studio, then select the Build

You will need to install the Windows USB driver in order to communicate with the Bioband, there is a user_guide.pdf file that explains the steps in the Windows directory

Automating the PC code

On Linux:

There is a simple shell script (bioband.sh) that will invoke the Demo program. It is easy to engineer the calling of this script upon a new Bioband being detected by the USB system by configuring a udev rule and placing that rule in /etc/udev/rules.d

For example ..

```
# udev config for bioband
ACTION=="add",SUBSYSTEM=="usb",ATTRS{idVendor}=="0483",ATTRS{idProduct}=="6000",
ATTRS{serial}=="?*",RUN+="/home/mrc/bioband2/udev/bioband.sh '$attr{serial}'"
```

Once added with pathname to the location of bioband.sh, refresh the udev rules by doing the following

```
sudo reload udev
```

On Windows:

At the current time, no in-depth work has been spent on automating the calling of the MRC or Demo programs by Windows detecting of USB event. If the Bioband work is taken forward this is a likely work area.

Toolchain

The band code needs to be cross compiled on the PC in order to run on the STM32 ARM Cortex-M3 processor. This section describes how to set up the cross compiler on a new/clean PC.

Note: creating the toolchain from scratch is a long winded process and it's likely that I've not covered every nuance below.

Do not proceed onto the next stage if the current stage you are executing fails to build as expected. In such circumstances it is best to delete the build directory and start again, be mindful of spelling mistakes and missing command parameters.

First off all set up the required base packages.

In a terminal

```
sudo apt-get install build-essential flex bison libgmp3-dev libmpfr-dev autoconf
sudo apt-get install texinfo libncurses5-dev libexpat1 libexpat1-dev


export TOOLPATH=/usr/local/cross-cortex-m3
sudo mkdir $TOOLPATH
```

```
sudo chown $USER:$USER $TOOLPATH
```

where $USER is your username. It is important to change the ownership since that ensures don't need to sudo in order to make install, which would require setting up the root user with the same environment.

Add this to your ~/.bashrc file

```
export TOOLPATH=/usr/local/cross-cortex-m3
export PATH=${TOOLPATH}/bin:$PATH
```

May also be worth setting up an alias to the bioband directory

```
alias bioband='cd /home/$USER/bioband2/software/Bioband'
```

Building the toolchain in your home directory ...

In the terminal enter the following

```
mkdir ~/toolchain
cd ~/toolchain

wget http://ftp.gnu.org/gnu/binutils/binutils-2.19.tar.bz2
tar -xvjf binutils-2.19.tar.bz2

wget http://fun-tech.se/toolchain/gcc/binutils-2.19_tc-arm.c.patch
patch binutils-2.19/gas/config/tc-arm.c binutils-2.19_tc-arm.c.patch

cd binutils-2.19
mkdir build
cd build
../configure --target=arm-none-eabi  \
             --prefix=$TOOLPATH  \
             --enable-interwork  \
             --enable-multilib  \
             --with-gnu-as  \
             --with-gnu-ld  \
             --disable-nls
make -j 2
make install

cd ~/toolchain
wget ftp://ftp.sunet.se/pub/gnu/gcc/releases/gcc-4.3.4/gcc-4.3.4.tar.bz2
tar -xvjf gcc-4.3.4.tar.bz2
cd gcc-4.3.4
mkdir build
cd build
../configure --target=arm-none-eabi  \
             --prefix=$TOOLPATH  \
             --enable-interwork  \
             --enable-multilib  \
             --enable-languages="c,c++"  \
             --with-newlib  \
             --without-headers  \
             --disable-shared  \
             --with-gnu-as  \
             --with-gnu-ld
make -j 2 all-gcc
make install-gcc
```

```
cd ~/toolchain
wget ftp://sources.redhat.com/pub/newlib/newlib-1.17.0.tar.gz
tar -xvzf newlib-1.17.0.tar.gz

wget http://fun-tech.se/toolchain/gcc/newlib-1.17.0-missing-makeinfo.patch
patch newlib-1.17.0/configure newlib-1.17.0-missing-makeinfo.patch

cd newlib-1.17.0
mkdir build
cd build
../configure --target=arm-none-eabi  \
             --prefix=$TOOLPATH  \
             --enable-interwork  \
             --disable-newlib-supplied-syscalls  \
             --with-gnu-ld  \
             --with-gnu-as  \
             --disable-shared

make -j 2 CFLAGS_FOR_TARGET="-ffunction-sections \
                        -fdata-sections \
                        -DPREFER_SIZE_OVER_SPEED \
                        -D__OPTIMIZE_SIZE__ \
                        -Os \
                        -fomit-frame-pointer \
                        -mcpu=cortex-m3 \
                        -mthumb \
                        -D__thumb2__ \
                        -D__BUFSIZ__=256" \
                        CCASFLAGS="-mcpu=cortex-m3 -mthumb -D__thumb2__"
make install


cd ~/toolchain
cd gcc-4.3.4/build
make -j 2 CFLAGS="-mcpu=cortex-m3 -mthumb" \
     CXXFLAGS="-mcpu=cortex-m3 -mthumb" \
     LIBCXXFLAGS="-mcpu=cortex-m3 -mthumb" \
     all
make install

cd ~/toolchain
wget http://ftp.gnu.org/gnu/gdb/gdb-7.0.tar.bz2
tar -xvjf gdb-7.0.tar.bz2
cd gdb-7.0
mkdir build
cd build
../configure --target=arm-none-eabi \
                      --prefix=$TOOLPATH  \
                      --enable-languages=c,c++ \
                      --enable-thumb \
                      --enable-interwork \
                      --enable-multilib \
                      --enable-tui \
                      --with-newlib \
                      --disable-werror \
                      --disable-libada \
                      --disable-libssp
make -j 2
make install
```