SIMtrace Usermanual

COLLABORATORS			
	TITLE : SIMtrace Usermanual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		April 7, 2012	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
0.0.1	12 July 2011	Initial	z
0.0.2	14 August 2011	Copy HW Info from the Wiki	z
0.0.3	15 August 2011	Document building wireshark	z
0.0.4	8 October 2011	Document Firmware	z
0.0.5	10 January 2012	Add additional distro packages, mention the v0.4 firmware update procedure, add some notes of the SAM-BA mode	z

Contents

List of Figures

Chapter 1

Introduction

1.1 History

SIMtrace was created out of necessity. Harald Welte wanted to see the communication between a GSM Mobile Station (or what we call a cellphone) and the SIM. He was not able to find an existing solution, or the existing ones had mayor drawbacks that made using them very time consuming and slow. The Atmel AT91SAM7 came to the rescue. This microcontroller has hardware support for the ISO7816 T0/T1 Smart Card specification. We can connect the external clock to the UART and are able to read bytes coming and going to the SIM. The next step in the project was taken by Kevin Redon that started to modify an existing AT91SAM7 design, started to use the Free Software KiCAD CAD Software. In 2011 the project went from having Schematics to having routed circuits, prototypes and the final product. The first production run was in August.

1.2 Overview

The setup of SIMtrace consists out of a Hardware and a Software part. The SIM card needs to be put into the SIMtrace Hardware, the flex cable needs to be connected to the SIMtrace Hardware and the SIM end needs to be placed in the SIM socket of the phone. The SIMtrace hardware can be seen as a USB device from the host, the SIMtrace software will try to find this device and claim it. The SIMtrace software will receive packets from the SIMtrace hardware and can forward them using the GSMTAP protocol to the IANA assigned GSMTAP port (4729). A modified version of Wireshark can be used to analyze the data.

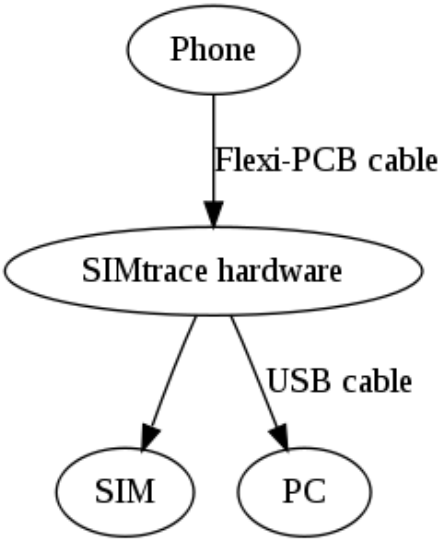


Figure 1.1: Schematic Overview

Chapter 2

Installation

SIMtrace will need a patched version of wireshark and the **simtrace** host utility to fully operate. The installation might be possible from binary packages or will require building from source. The following sections provide some hints how to achieve this on the various Linux distributions. All these operations must be executed as root.

Wireshark Patches
The SIMtrace patch has been upstreamed. Wireshark 1.7.1 was released on April the 6th 2012 and is the first development release to support SIMtrace out of the box. Wireshark 1.8 will be the first stable version.

2.1 Installation Ubuntu Natty, Ubuntu Oneiric

Ubuntu Natty and Oneiric users can use the holger+lp/osmocom PPA to install SIMtrace and upgrade wireshark. The PPA needs to be added to the system, the package database needs to be refreshed and the applications can be installed afterwards.

```
$ sudo add-apt-repository ppa:holger+lp/osmocom
[sudo] password for username:
Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --secret-keyring / ↩
    etc/apt/secring.gpg --trustdb-name /etc/apt/trustdb.gpg --keyring /etc/apt/trusted.gpg  ↩
    --primary-keyring /etc/apt/trusted.gpg --keyserver hkp://keyserver.ubuntu.com:80/ --recv ↩
     84C86214C00BAF820F43585CCABF944FA2AD19FA
gpg: requesting key A2AD19FA from hkp server keyserver.ubuntu.com
gpg: Total number processed: 1
gpg:              unchanged: 1
```

The next step is to update the package database and install or upgrade the wireshark application.

```
$ sudo apt-get update
...
$ sudo apt-get install wireshark simtrace
...
```

Note
The wireshark will only be installed if it is newer than the version provided by Ubuntu. Please verify that the above command installed a version coming from the PPA.

2.2 Installation OpenSUSE

The installation on OpenSUSE uses zypper. The repository must be added via the **zypper** application and then the binary packages can be installed.

2.2.1 openSUSE 11.3

```
$ zypper addrepo http://download.opensuse.org/repositories/home:/zecke23/openSUSE_11.3/home ↩
    :zecke23.repo
$ zypper refresh
$ zypper install wireshark simtrace
```

2.2.2 openSUSE 11.4

```
$ zypper addrepo http://download.opensuse.org/repositories/home:/zecke23/openSUSE_11.4/home ↩
    :zecke23.repo
$ zypper refresh
$ zypper install wireshark simtrace
```

2.2.3 openSUSE 12.1

```
$ zypper addrepo http://download.opensuse.org/repositories/home:/zecke23/openSUSE_11.4/home ↩
    :zecke23.repo
$ zypper refresh
$ zypper install wireshark simtrace
```

2.3 Installation Fedora

2.3.1 Fedora 14

```
$ cd /etc/yum/repos.d/
$ wget http://download.opensuse.org/repositories/home:zecke23/Fedora_14/home:zecke23.repo
$ yum install wireshark simtrace
```

2.3.2 Fedora 15

```
$ cd /etc/yum/repos.d/
$ wget http://download.opensuse.org/repositories/home:zecke23/Fedora_15/home:zecke23.repo
$ yum install wireshark simtrace
```

2.3.3 Fedora 16

```
$ cd /etc/yum/repos.d/
$ wget http://download.opensuse.org/repositories/home:zecke23/Fedora_16/home:zecke23.repo
$ yum install wireshark simtrace
```

2.4 Installation CentOS

2.4.1 CentOS 5

```
$ cd /etc/yum/repos.d/
$ wget http://download.opensuse.org/repositories/home:zecke23/CentOS_CentOS-5/home:zecke23. ↩
    repo
$ yum install wireshark simtrace
```

2.4.2 CentOS 6

```
$ cd /etc/yum/repos.d/
$ wget http://download.opensuse.org/repositories/home:zecke23/CentOS_CentOS-6/home:zecke23. ↩
    repo
$ yum install wireshark simtrace
```

2.5 Mandriva

2.5.1 Mandriva 2010.1

```
$ urpmi.addmedia home:zecke23 http://download.opensuse.org/repositories/home:zecke23/ ↩
    Mandriva_2010.1/
$ urpmi.update -a
$ urpmi wireshark simtrace
```

2.5.2 Mandriva 2011

```
$ urpmi.addmedia home:zecke23 http://download.opensuse.org/repositories/home:zecke23/ ↩
    Mandriva_2011/
$ urpmi.update -a
$ urpmi wireshark simtrace
```

2.6 Installation from Source

Please see the Chapter 5

Chapter 3

Hardware Details

3.1 HW Design

The Free Software KiCAD EDA was used to design the hardware and can be used to look at the schematics and the PCB routing. The hardware design can be found in the git repository of the SIMtrace sources. For the v1.0 hardware you will have to look at the v1.0_production branch.

3.2 Populated PCB

The version v1.0p is the first production that had an automatic SMT run. Due some production issues the labeling of components didn't make it to the PCB but can be found in this manual. The difference between the v1.0 and v1.0p hardware is in the footprint of some components to utilize the existing stock of the factory. This was mostly done for the LED and the shottky diodes.

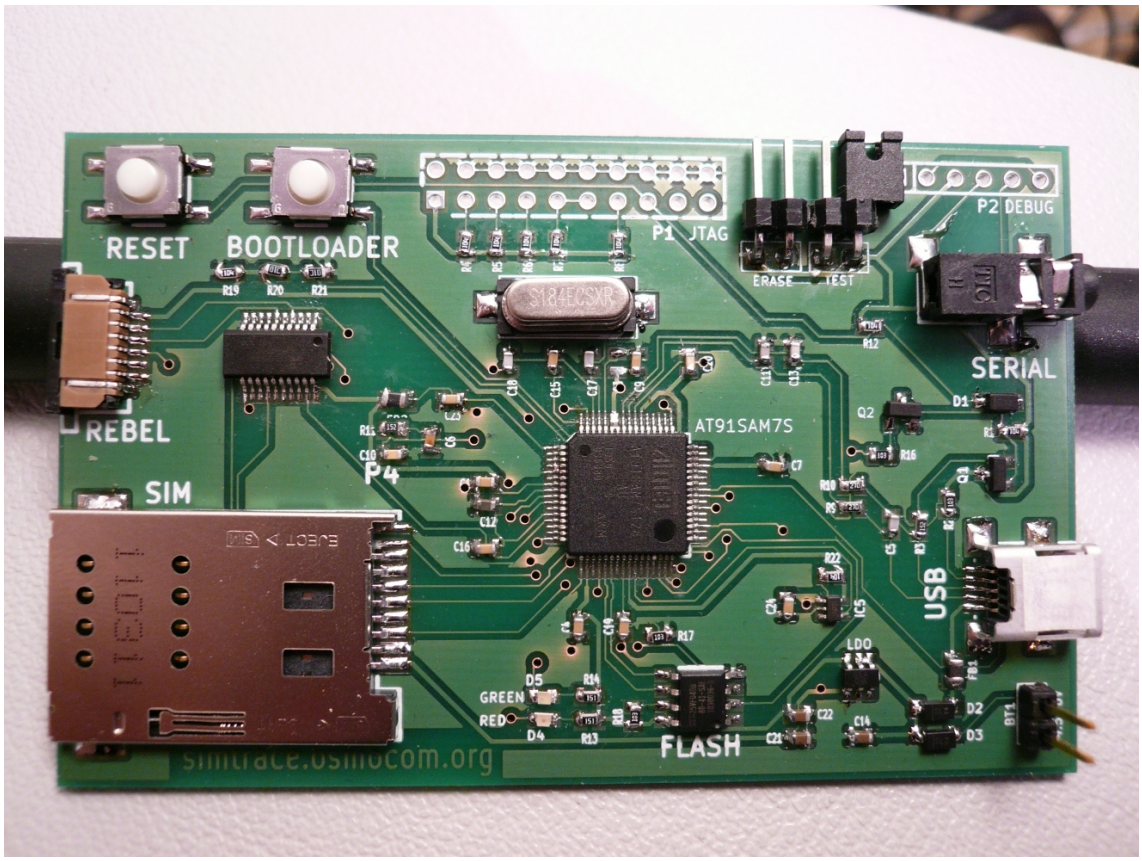
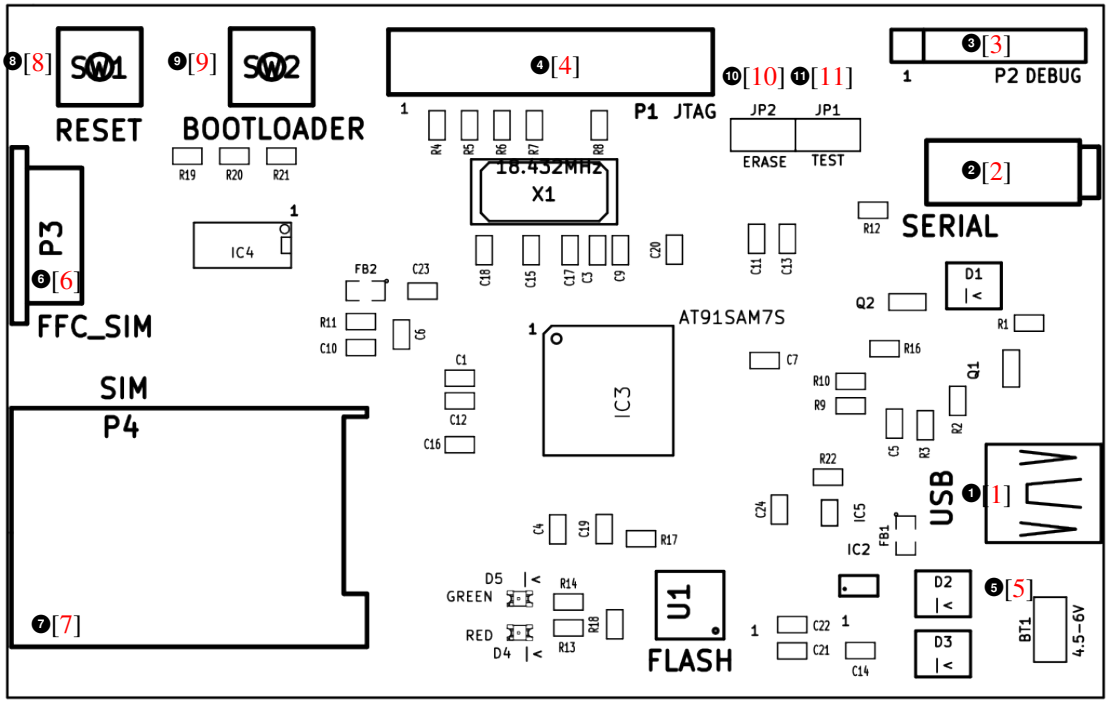


Figure 3.1: SIMtrace v1.0 PCB

3.3 PCB Surface



1. USB: USB mini-B connector. The main connector. The host software communicates (sniffing,...) through USB with the board. It can also be used to flash the micro-controller (using DFU).
2. serial: 2.5 mm jack serial cable, as used by osmocomBB port used to debug the device (printf goes there).
3. debug (P3): same as serial, but using the FTDI serial cable. It is recommended to cut the voltage wire of the 6pin FTDI connector before plugging the cable into the simtrace.
4. jtag (P1): JTAG 20 pin connector to do hardware assisted debugging.
5. BT1: battery connector (4.5-6V DC). normally the USB provides power, but the battery port can be used for autonomous use of SIMtrace. The sniffing can be saved in the flash (U1).
6. FFC_SIM (P3): to connect the flat flexible cable with SIM end for the phone.
7. SIM (P4): put your SIM in there (instead of in the phone)
8. reset (SW1): to reset the board (not erasing the firmware). If your are too lazy to unplug and re-plug the USB.
9. bootloader (SW2): used to start the bootloader so to flash the device using DFU. press when plugging in the USB.
10. test (JP1): short circuit using a jumper to flash using SAM-BA.
11. erase (JP2): short circuit using a jumper to erase completely erase the firmware.

Figure 3.2: SIMtrace v1.0 Surface

Chapter 4

Sniffing your SIM

4.1 Connecting your device

You will need to put your SIM into the SIMtrace hardware, connect one of the four flex cables to the SIMtrace hardware, put the other side into the SIM socket of your phone. Use USB to connect the SIMtrace hardware to the PC. On your PC you should be able to see the USB device now.

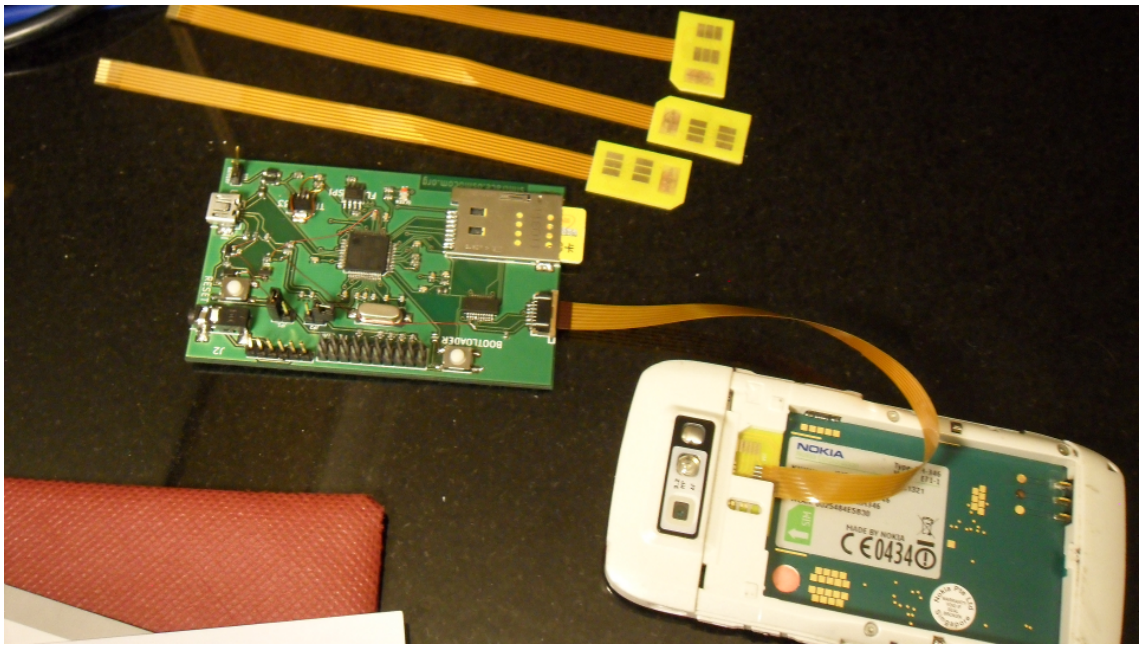


Figure 4.1: Connecting the SIMtrace Hardware

4.2 Launching SIMtrace

```
$ ./simtrace
simtrace - GSM SIM and smartcard tracing
(C) 2010 by Harald Welte <laforge@gnumonks.org>
```

Launching the **simtrace** will try to find the SIMtrace hardware and then try to claim the USB device. The application will send the received data encapsulated in the GSMTAP format on localhost and the IANA assigned GSMTAP port.

4.3 Launching Wireshark

The **wireshark** application will start a GUI and given the right permissions you should be able listen to the localhost interface and filter for the GSMTAP port on 4729. You should be able to see the decoded messages like in the figure below.

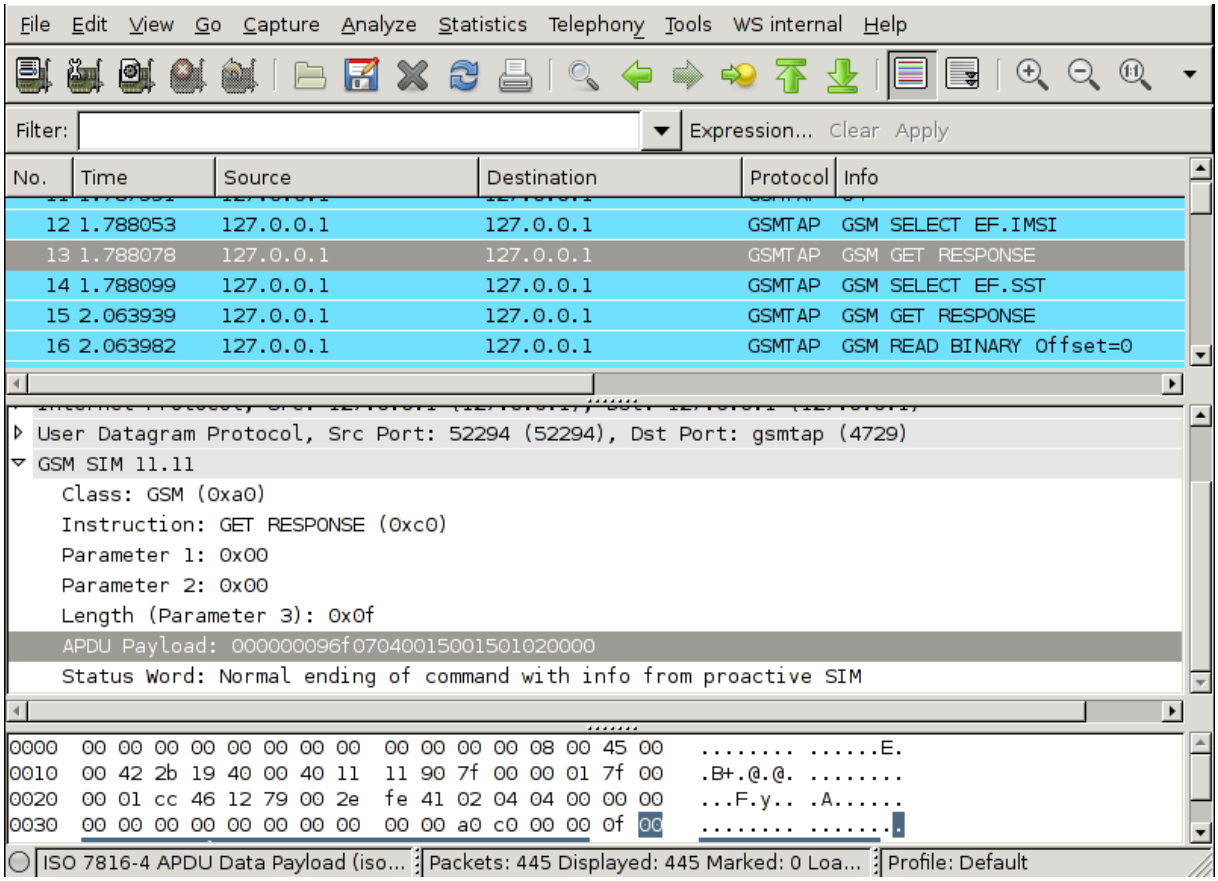


Figure 4.2: GSMTAP in Wireshark

4.4 Known Firmware Issues

4.4.1 Combined ATR/APDU message

For some cards the firmware does not send an USB message at the end of the ATR. The ATR and first APDU will be send in one message and the host utility fails to split APDUs and nothing will be traced. A band-aid for the firmware exists and can be found on the mailinglist.

4.4.2 Lost bytes

For some new high speed cards the firmware can lose bytes. The issue appears to be when the received bytes will be copied to the memory of the USB controller. The workaround is to reduce the size of the buffer.

4.5 Other modes

The hardware is capable to be used as an ordinary card reader, provide Man-In-The-Middle (MITM) attacks, or operate as a SIM. The firmware currently does not have support for these modes.

The SIMtrace hardware supports ISO7816 Part 3 T=0/T=1 protocols, it basically can be used to intercept and analyze any traffic from (ISO7816) smart cards. This includes SIM cards, Pay TV cards (smart card for CAM), ATM cards, chip credit card, PKI smart cards, e-passport etc. etc. However watch out: You have to make your chip card fitting in the "SIM card size" ID-000 reader or build another adapter.

Chapter 5

Getting and Building the Software

5.1 Building software

There are three parts that can be built. It is the firmware for the SIMtrace hardware, the SIMtrace software and the modified version of wireshark. All of these have different source trees and dependencies.

5.2 Building SIMtrace

5.2.1 Building the Osmocom libosmocore library

```
$ git clone git://git.osmocom.org/libosmocore
$ cd libosmocore
$ autoreconf --install --force
$ ./configure
$ sudo make install
```

5.2.2 Installing libusb

You will need to install the libusb header files to be able to compile **simtrace**.

5.2.3 Building simtrace

```
$ wget https://api.opensuse.org/public/source/home:zecke23/simtrace/simtrace_0.0.1.tar.gz
$ tar xzf simtrace_0.0.1.tar.gz
$ cd simtrace-0.0.1
$ PKG_CONFIG_PATH=/usr/local/lib/pkgconfig make
cc  `pkg-config --cflags libosmocore` -o main.o -c main.c
cc  `pkg-config --cflags libosmocore` -o usb_helper.o -c usb_helper.c
cc  `pkg-config --cflags libosmocore` -o usb.o -c usb.c
cc  `pkg-config --cflags libosmocore` -o apdu_split.o -c apdu_split.c
cc -o simtrace main.o usb_helper.o usb.o apdu_split.o -lusb `pkg-config --libs libosmocore` ↩
    -losmocore
```

5.3 Building Wireshark

SIMtrace provides a patch against **wireshark** version 1.6. It is the easiest to checkout a copy using the 1.6 branch of wireshark and applying the `simcard.patch` on top of it. And then use the usual way of building wireshark

5.3.1 Getting Wireshark

```
$ svn co https://anonsvn.wireshark.org/wireshark/trunk-1.6 wireshark-1.6
...
A    wireshark-1.6/isprint.h
 U   wireshark-1.6
Checked out revision 38543.
```

5.3.2 SIMCard patch

You will need to download and apply the patch.

```
$ cd wireshark-1.6
$ wget http://cgit.osmocom.org/cgit/simtrace/tree/wireshark/simcard-for-wireshark-1.6.patch
$ cat ../simcard-for-wireshark-1.6.patch | patch -p 0
patching file epan/dissectors/packet-gsm_sim.c
patching file epan/dissectors/packet-gsmtap.c
patching file epan/dissectors/Makefile.common
```

5.3.3 Building and Installing

```
$ autoreconf --install
$ ./configure
$ make
...
$ sudo ./wireshark
```

Chapter 6

Getting and Building the Firmware

6.1 Introduction

The Firmware is the Software that is running on the Microcontroller of the SIMtrace hardware. The Firmware itself consists out of a couple of components for different parts of the system. Besides the source code for the firmware you will also need to have an ARM Cross-Compile Toolchain, a copy of the SAM7 utilities to initially program the device or recover from a fatal error and dfu-util to update the main part of the firmware using the Device Firmware Update (DFU) mode.

6.2 Getting a Toolchain

The toolchain needs to include a GCC newer than 3.4 and it may not be an EABI toolchain. EABI toolchains fail to properly link the SIMtrace binary. You can easily build a toolchain yourself or use one of the known working pre-built ones. Please see the SIMtrace wiki for more information about getting a toolchain.

6.3 Getting and Building the Firmware

The SIMtrace firmware is based on the OpenPCD RFID Reader Firmware and the SIMtrace firmware code is located in the OpenPCD repository. You can use the **git** to clone the repository.

```
$ git clone git://git.gnumonks.org/openpcd.git
```

The firmware consists out of two separate binaries that will be concatted and flashed into the NOR flash of the microcontroller. The main part is the dfu program that will handle basic USB functionality and respond to Device Firmware Update (DFU) requests to allow to update the firmware in the NOR or execute software in RAM.

```
$ cd openpcd/firmware
$ make -f Makefile.dfu BOARD=SIMTRACE
$ make BOARD=SIMTRACE DEBUG=1 TARGET=main_simtrace
$ cat dfu.bin main_simtrace.bin > main_simtrace.samba
$ cd ../..
```

6.4 Firmware Details

The handling for the DFU part can be found in the `src/dfu` directory, it also provides low-level USB routines to work with USB Device Port (UDP). These functions will be called from the main payload.

The operating system part is in `src/os` it provides basic hardware control and services to be used by the main application, this includes USB enumeration, Watchdog programming, running the mainloop, interrupt dispatching. The main application for SIMtrace can be found in `src/simtrace` and this includes programming the two USART, configuring the bus switch according to the mode.

6.5 Initial Firmware Programming

In case the NOR Flash of the SAM7 Microcontroller is either blank or has become corrupted the Microcontrollers support entering a mode called SAM-BA which then allows flashing the device using the `sam` application. The SAM-BA mode can be easily entered by following the below procedure. ENTERING SAM-BA MODE

1. Unplug the SIMtrace Hardware from USB.
2. Short TEST to VCC (3.3V) pin by using the Jumper. Leave PA0, PA1, PA2 unconnected.
3. Power up the SIMtrace Hardware from USB.
4. Wait for 20 seconds.
5. Unplug the SIMtrace Hardware from USB.
6. Open/Remove the Jumper.

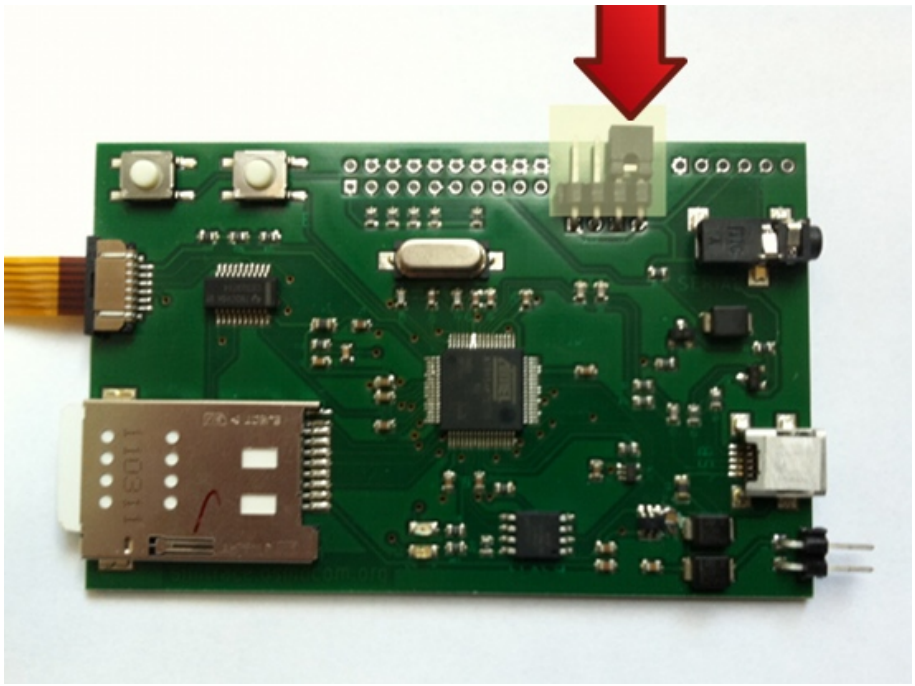


Figure 6.1: TEST Jumper

v1.0p/v1.1p Hardware Owners
Sometimes the SAM-BA mode is not entered. This is the case when the two LEDs are on when powering up the SIMtrace Hardware with the Jumper set. The reason for this is unknown but there are several workarounds:

- Press the RESET button while powering up.
- In addition, remove the jumper and put it back.

As soon as the two LEDs go off, the SAM-BA mode is properly entered.

The **sam** application can be compiled to either use libusb or normal files to program the device, depending on the drivers used you will need to configure the application one way or another. The programming can then be done using the below command.

```
$ ./sam7 --exec set_clock --exec unlock_regions --exec "flash ../openpcd/firmware/ ↩
    main_simtrace.samba"
```

Silent failures
The **sam** can silently fail when not finding or being able to configure the device properly. It is best to enter the interactive mode by not providing any **--exec** commands.

6.6 Device Firmware Update

The Device Firmware Update (DFU) part of the firmware will be booted first, it is checking if a button is active or if the software reset reason was for DFU and then activates the DFU part or jumps to the main application. DFU can be activated at any time using **dfu-util** on the USB Host.

The **dfu-util** application might be already packaged for your distribution, the source code can be found on the dfu-util.gnumonks.org website. To update the main part of the firmware simply do:

```
$ dfu-util -d 16c0:0762 -a0 -D ./main_simtrace.bin -R
```

6.7 Upgrading to v0.4 Firmware

Upgrading to v0.4 requires flashing both the Bootloader and the SIMtrace application. The procedure is first to flash the bootloader, then the SIMtrace application and finally reset the device.

```
$ dfu-util -d 16c0:0762 -a 1  -D ./dfu.bin
$ dfu-util -d 16c0:0762 -a 0  -D ./main_simtrace.bin
... reset the device
```