# Using STM32 discovery kits with open source tools

STLINK development team

# Contents

# 1 Overview

This guide details the use of STMicroelectronics STM32 discovery kits in an opensource environment.

## 2 Installing a GNU toolchain

Any toolchain supporting the cortex m3 should do. You can find the necessary to install such a toolchain here:

```
https://github.com/esden/summon-arm-toolchain
```

Details for the installation are provided in the topmost README file. This documentation assumes the toolchains is installed in a $TOOLCHAIN_PATH.

# 3 Installing STLINK

STLINK is an opensource software to program and debug the discovery kits. Those kits have an onboard chip that translates USB commands sent by the host PC into JTAG commands. This chip is called STLINK, which is confusing since the software has the same name. It comes into 2 versions (STLINK v1 and v2). From a software point of view, those versions differ only in the transport layer used to communicate (v1 uses SCSI passthru commands, while v2 uses raw USB).

Before continuing, the following dependencies are required:

- libusb-1.0
- libsg2

The STLINK software source code is retrieved using:

```
git clone https://github.com/texane/stlink stlink.git
```

The GDB server is called st-util and is built using:

```
$> cd stlink.git;
$> make ;
$> cd gdbserver ;
$> make ;
```

# 4 Building and running a program

A simple LED blinking example is provided in the example directory. It is built using:

```
cd stlink.git/example/blink ;
PATH=$TOOLCHAIN_PATH/bin:$PATH make ;
```

A GDB server must be start to interact with the STM32. Depending on the discovery kit you are using, you must run one of the 2 commands:

```
# STM32VL discovery kit
$> sudo ./st-util /dev/sg2

# STM32L discovery kit
$> sudo ./st-util
```

Then, GDB can be used to interact with the kit:

```
$> $TOOLCHAIN_PATH/bin/arm-none-eabi-gdb
```

From GDB, connect to the server using:

```
$> target extended localhost:4242
```

By default, the program was linked such that the base address is 0x20000000. From the architecture memory map, GDB knows this address belongs to SRAM. To load the program in SRAM, simply use:

```
$> load blink.elf
```

GDB automatically set the PC register to the correct value, 0x20000000 in this case. Then, you can run the program using:

```
$> continue
```

The board BLUE and GREEN leds should be blinking (those leds are near the user and reset buttons).

# 5 Notes

By default, the disassemble command in GDB operates in ARM mode. The programs running on CORTEX-M3 are compiled in THUMB mode. To correctly disassemble them under GDB, uses an odd address. For instance, if you want to disassemble the code at 0x20000000, use:

```
$> disassemble 0x20000001
```

# 6 References

- http://www.st.com/internet/mcu/product/248823.jsp
  documentation related to the STM32L mcu
- http://www.st.com/internet/evalboard/product/250990.jsp
  documentation related to the STM32L discovery kit