PC prerequisites

Contents

Stable: 25.03.2021 - 16:36 / Revision: 25.03.2021 - 16:36

Stable: 25.03.2021 - 10:00 / Revision: 25.03.2021 - 10:00

A quality version of this page, approved on *25 March 2021*, was based off this revision.

Information

Recommended setup: Native Linux® PC

Contents

1 Purpose

This article explains and describes the host PC hardware and software configuration required to activate and run the STM32 MPU platforms.

2 Recommended PC configurations

The PC requirements depend on the Package you want to use.

The table below guides through the selection and configuration of the host PC environment according the targeted Package:

Host environment	Starter Package	Developer Package	Distribution Package
Windows® (64 bits) Tested with Windows 7 and Windows 10 Preferred version is Windows 10	native	Virtual machine	Virtual Machine
Linux (64 bits) Tested with Ubuntu® **18.04** and 20.04	Native	Native + additional packages (see Linux PC chapter)	native + additional packages (see Linux PC chapter)

There are no absolute minimal requirements regarding the PC hardware configuration. However ST recommends to meet or exceed the following hardware requirements when using the *Developer Package* or *Distribution Package*.

The table below corresponds to the minimal validated configuration:

Hardware item	Minimal validated configuration	Comments / Recommendations
CPU	core i5-2540M @ 2.6 GHz 2 cores (4 threads) 3-Mbyte cache	**64-bit instruction set is mandatory** **8 cores/threads or more** is a good configuration for *Developer Package* and *Distribution Package*.
RAM	8 Gbytes	**16 Gbytes or more** are recommended especially for *Virtual machine* setup, *Developer Package* and *Distribution Package*.
Hard drive	320 Gbytes	**1 Tbytes** is the best suited configuration for *Distribution Package*

3 Linux® PC

A Linux PC with **Ubuntu 18.04** is the recommended setup. Other Ubuntu revisions are also supported (refer to Yocto reference manual[1]) .

Information

ecosystem release v2.1.0

ST solutions are tested and validated on a Linux PC running Ubuntu 18.04 LTS (and Ubuntu 20.04 LTS).

Information

ecosystem release v2.0.0

ST solutions are tested and validated on a Linux PC running Ubuntu 18.04 LTS (and Ubuntu 16.04 LTS).

3.1 Checking Internet access

- Internet access through http and https protocols

Required at least for *Developer Package* and *Distribution Package*.

The command below enables checking internet access through http/https protocols:

```
PC $> wget -q www.google.com && echo "Internet access over HTTP/HTTPS is OK !" || echo
"No internet access over HTTP/HTTPS ! You might need to set up a proxy."
```

If an 'OK' message is returned, the network is well configured. In such case, skip the rest of this section and jump to Install extra packages.

Any other situation likely indicates that a proxy is required for http/https protocols. The best solution to set a proxy for http/https protocols is via the shell variables http_proxy and https_proxy:

```
PC $> export http_proxy=http://<MyProxyLogin>:<MyProxyPassword>@<MyProxyServerUrl>:<MyProx
yPort>
PC $> export https_proxy=http://<MyProxyLogin>:<MyProxyPassword>@<MyProxyServerUrl>:<MyPro
xyPort>
```

Since your password (<MyProxyPassword>) might contain "special characters", translate it into ASCII hexacode. To do this, use the command below:

PC $> echo -n " <MyProxyPassword>" | od -A n -t x1 -w128 | head -1 | tr " " "%"

Check again the Internet access by using the following command:

```
PC $> wget -q www.google.com && echo "Internet access over HTTP/HTTPS is OK !" || echo
"No internet access over HTTP/HTTPS ! You might need to set up a proxy."
```

- **Internet access for *sudo* commands**

Required for *Distribution Package*.

By default, *sudo* commands are executed in the *root* user environment; no Internet proxy settings are applied for *root* users. *Root* users must be able to browse Internet after having created an alias passing the proxy settings on *sudo* commands:

```
PC $> alias sudo='sudo http_proxy=$http_proxy'
```

Check that the *sudo* commands are successful (requires Internet access):

```
PC $> sudo apt-get update
```

- **Internet over git://, ssh:// and others specifics protocols**

Required for *Distribution Package*.
In addition to http/https protocols (used in 90% of the Internet traffic), other protocols such as git:// or ssh:// might be required.

For example, in the context of the *Distribution Package*, some "git fetch" commands might require "git:// protocols".
To support these protocols through a proxy, directly setup the proxy in the $HOME/.gitconfig file (core.gitproxy) and use a tool such as *cockscrew*[2] to tunnel the git:// flow into the http flow:

```
PC $> sudo apt-get update
PC $> sudo apt-get install corkscrew

PC $> git config --replace-all --global core.gitproxy "$HOME/bin/git-proxy.sh"
PC $> git config --add --global core.gitproxy "none for <MyPrivateNetworkDomain>"
(optional, for example .st.com or localhost)
PC $> echo 'exec corkscrew <MyProxyServerUrl> <MyProxyPort> $* $HOME/.git-proxy.auth' >
$HOME/bin/git-proxy.sh
PC $> chmod 700 $HOME/bin/git-proxy.sh
PC $> echo '<MyProxyLogin>:<MyProxyPassword>' > $HOME/.git-proxy.auth
PC $> chmod 600 $HOME/.git-proxy.auth
```

Use the command below to test the proxy settings:

```
PC $> git ls-remote git://git.openembedded.org/openembedded-core > /dev/null && echo OK
|| echo KO
```

The command returns 'OK' if the proxy settings are correct.

3.2 Installing extra packages

Required for *Developer Package* and *Distribution Package*.

Additional Ubuntu packages must be installed to perform basic development tasks, basic cross-compilation (via *Developer Package*) or more complex cross-compilation such as OpenEmbedded does (via *Distribution Package*):

- Packages required by OpenEmbedded/Yocto (details here):

```
PC $> sudo apt-get update
PC $> sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-
essential chrpath socat cpio python3 python3-pip python3-pexpect xz-utils debianutils
iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3 pylint xterm
PC $> sudo apt-get install make xsltproc docbook-utils fop dblatex xmlto
```

- Packages needed for some "Developer Package" use cases:

```
PC $> sudo apt-get install libncurses5 libncurses5-dev libncursesw5-dev libssl-dev linux-
headers-generic u-boot-tools device-tree-compiler bison flex g++ libyaml-dev
```

- Package for repo (used to download the "Distribution Package" source code):

```
Please follow the installation instructions described in repo.
 For Ubuntu 16.04, use the legacy repo old-repo-python2 chapter
```

- Useful tools:

```
PC $> sudo apt-get install coreutils bsdmainutils sed curl bc lrzsz corkscrew cvs
subversion mercurial nfs-common nfs-kernel-server libarchive-zip-perl dos2unix texi2html
diffstat libxml2-utils
```

You can also install a Java Runtime Engine that is required for STM32CubeMX and STM32CubeProgrammer

```
PC $> sudo apt-get install default-jre
```

3.2.1 Installing extra packages for Android™

Below information is related to the Android™ distribution

Before downloading and building the STM32MPU distribution for Android™, make sure your system meets the following requirements:

- A 64-bit Linux® environment.
- At least 150 Gbytes of free disk space. If you conduct multiple builds, even more space is required.
- At least 8 Gbytes of RAM/swap. If you are running Linux on a virtual machine, at least 16 Gbytes of RAM/swap are required.

For more details, refer to the requirements page of the AOSP website.

Use the following commands to install the packages required to build an environment for Android (Distribution Package), after having installed the required packages listed on Android website:

```
sudo apt-get update
sudo apt-get install chrpath curl libxml2-utils gdisk pv python-
pycryptopp python-crypto autotools-dev automake libusb-1.0-0-dev
```

To ensure USB communication (with ADB) between the host and the device, see ADB § Device Connection.

To run Android-provided tests (CTS and VTS), see Android Platform Testing.

At this stage: Your environment is ready for Android build, debug and test.

3.3 Additional configurations

- Allowing up to 16 partitions per MMC

By default a maximum of 8 MMC partitions are allowed on Linux systems. All Packages (such as Starter Package) need more than 10 partitons for the storage device. To extend the number of partitions per device to 16, the following options must be added to modprobe:

```
PC $> echo 'options mmc_block perdev_minors=16' > /tmp/mmc_block.conf
PC $> sudo mv /tmp/mmc_block.conf /etc/modprobe.d/mmc_block.conf
```

- Checking for *locale* setup

Required for *Distribution Package*.

The *locale* setting is used by some applications/commands (including by *Distribution Package* applications/commands).
Verify that the *locale* settings are as follows:

```
PC $> locale
LANG=en-US.UTF-8
```

If the *locale* command returns a different configuration than the one shown above, reconfigure it as follows:

```
PC $> sudo update-locale LANG=en_US.UTF-8
```

- Adding users in basics groups

The *user* login must belong to the basic Linux groups such as **disk**, **tty**, **dialout** or **plugdev**
Use the *groups* command to list groups for the current user:

```
PC $> groups
```

If needed, add *user* to the missing *<groups>*:

```
PC $> sudo adduser $USER <group>
```

Then **reboot** the PC.

3.4 Seting up *Git* user information

Required for *Developer Package* and *Distribution Package*.

The user Information are needed by git[3] if *commit* and/or *push* commands are used:

```
PC $> git config --global user.name "Your Name"
PC $> git config --global user.email "you@example.com"
```

4 Windows PC

Starter Package **may run on Windows.**
Developer Package **and** ***Distribution Package*** **require a Linux environment.**

Warning

ST solutions, while reportedly functional when running on a Linux Virtual machine, are only validated for Linux native setups.

There are several ways to run Linux system on top of a Windows host PC. ST recommends to use a Virtual Machine System:

1. Install a virtual machine such as VMWare [4]
2. Setup a **64-bit** Ubuntu image compatible with your virtual machine

ST, in an experimental way, has also run *Developer Package* and *Distribution Package* on WSL2 (Windows Subsystem for Linux 2). Refer to WSL2 chapter.

4.1 Virtual machine system

4.1.1 Installing the virtual machine

ST has selected VMWare as Linux virtual machine solution.

VMWare is a commercial company specialized in virtualization solutions. The available solutions to support a virtual Linux machine on a Windows PC are:

- VMWare Workstation Player (paid solution) for commercial use (download here [5])
- VMWare Workstation Player (free solution) for home use (download here [6])

Please proceed with the installation of the virtual machine.

Before running the virtual machine, make sure the virtualization is activated in the BIOS (it should be activated by default for any retail PC).

4.1.2 Downloading the Ubuntu image for the virtual machine

The "osboxes.org" [7] website provides virtual machine images compatible with VMWare(*.vmdk).

ecosystem release v2.1.0 : **Setup have been validated and tested on Ubuntu 18.04 (64bits)** and Ubuntu 20.04 (64bits).
ecosystem release v2.0.0 : **Setup have been validated and tested on Ubuntu 18.04 (64bits)** and Ubuntu 16.04 (64bits).

Download the 64-bit Ubuntu image available at [8], then:

1. Unzip the downloaded file
2. In VMware, create a virtual machine using the Ubuntu virtual disk downloaded from osboxes.org.

The recommended usage is to dedicate, at least, half of the host machine to the virtual machine:

```
- CPU: 2 cores at least,
- RAM: 6 Gbytes or more is a good choice (the more RAM allocated to Virtual Machine the
better - the RAM allocated to Virtual Machine must be at least 4 Gbytes),
- Network: NAT is a good easy way to benefit from a network connection within the virtual
machine.
```

The virtual size of the virtual disk downloaded from osboxes.org is about 500 Gbytes. Even if, at beginning, the real size of the file of the virtual disk is less, the size can grow up to 500 Gbytes over distribution package compiling or package development.

Information

For VMware, first create a default virtual machine, then add the previously downloaded *.vmdk* file.

Please refer to the VMwarePlayer screenshot tutorial.

4.1.3 Launching Ubuntu image

Warning

For "AZERTY" keyboard users:

The default keyboard configuration is "QWERTY".
To configure the keyboard to "AZERTY", start by opening a session (take care that the keyboard layout is QWERTY).
TIP: the password for the default user "*osboxes.org*" is "*osboxes.org*".
TIP: the '.' character is obtained by clicking ':' on an AZERTY keyboard configured in QWERTY.
Once the session is open, click the 'En' icon on top/right of the screen, select the French ('Fr') keyboard layout and move it to the first position in the list.
Optionally the 'En' keyboard can be completely removed. If the 'Fr' option is not present, it can be added with the 'Text entry setting' menu.

Default 'Ubuntu *Credentials'* **are set to "osboxes.org" for both login and password.**

Warning

Adjusting screen resolution:

The (default) resolution used by the virtual machine is 800x600 (smallest available). It is not automatically adjusted to the display resolution. To adjust the resolution, click the "settings" icon ('toothed wheel' on top/right of the screen), then "system settings ..." > "display" and select the appropriate resolution for the display (do not to forget to click the "Apply" button on bottom/left of the "Screen Resolution Setting" window).

For a better experience with the VMware virtual machine, install "vmware-tools" in order to be able to use the clipboard to drag-and-drop and copy/paste files between VMware and Windows. A step-by-step installation procedure of **vmware-tools** is available in PreRequisite-Vmware-tools.pdf.

The virtual machine is now up and running!

The Ubuntu setup must be finalized according recommendations provided in Linux PC chapter

Warning

USB connection speed:

A USB connection is required to access STLink (debugger and serial port) and by STM32CubeProgrammer. The speed of the USB connection between Linux running in the virtual machine and the external USB devices can be severely impacted by the following factors:

- the virtual machine USB setup
- the USB controller in the host PC
- the USB device connected to host PC
- any USB hub between the USB host and the USB device.

If the speed of your USB connection is too low:

- try different USB configurations of the virtual machine;
- connect the USB device directly to the host USB port (without any USB hub);
- try connecting the USB device to another USB port of the host (some PC have different USB controller on different USB port).

4.2 WSL2 (experimental)

Even if STMicroelectronics strongly recommends to use a Linux® environment, the *Developer Package* and *Distribution Package* work on WSL2 (Windows Sub-system Linux 2) environment. WSL is a feature provided by **Windows 10®**.
WSL2 is a new version of the architecture that powers the Windows subsystem for Linux to run ELF64 Linux binaries on Windows (more details on aka.ms/wsl2). It is available on Windows 10 since build 18917. The *Developer Package* successfully runs on WSL2 but unsuccessfully on WSL.

- Installing WSL2 :
  - To install WSL2, read https://docs.microsoft.com/fr-fr/windows/wsl/wsl2-install
  - Once WSL2 is installed, jump to #Linux_PC to make your WSL2 ready to run *Developer Package* and/or *Distribution Package*.
- WSL2 limitations :
  - Up to now (September 2019), WSL2 does not support hardware such as USB or serial devices (more details).
    This means that STM32CubeProgrammer must be used through native Windows.
  - WSL2 files are not browsable from Windows native file explorer.
    To share files between WSL2 and Windows, it is recommend to use the mount point */mnt/c* from WSL2 and perform copies.
- Tips for using WSL2 :
  - Launching graphical application : on *wiki.ubuntu.com*, go to WSL page and read the chapter Running Graphical Applications.

4.3 References

- Supported Linux Distributionsl
- https://en.wikipedia.org/wiki/Corkscrew_(program)
- Git
- http://vmware.com
- https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/15_0
- https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html
- http://osboxes.org
- https://www.osboxes.org/ubuntu/#ubuntu-1804-vmware

Linux® is a registered trademark of Linus Torvalds.

Microprocessor Unit

Central processing unit

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today for volatile random-acces ssemiconductor memories.)

Android Open Source Project

Android debug bridge (Android specific)

Compatibility Test Suite (Android specific) or Clear to send (in UART context)

Vendor Test Suite (Android specific)

MultimediaCard

ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families (See ST-LINK for more details)

Stable: 15.02.2021 - 12:33 / Revision: 12.02.2021 - 08:25

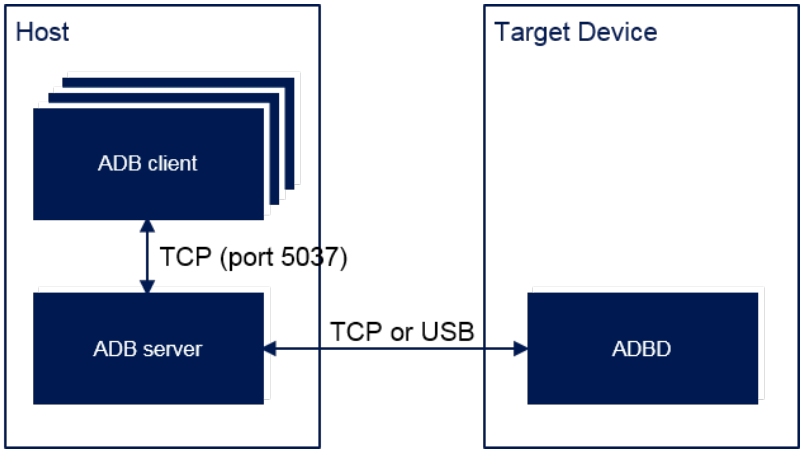
A quality version of this page, approved on *15 February 2021*, was based off this revision.

The Android Debug Bridge (ADB) is a versatile command-line tool that lets you communicate with a device (an emulator or a connected Android device).

This article is intended for Developer Package or Distribution Package users (see Which Package better suits your needs for more information).

Contents

1 Overview



- The **ADB client** is an executable that can be launched with a subcommand such as `adb shell` or `adb logcat`.
- The **ADB server** acts as a proxy between adb clients and adbd.
- The **ADBD daemon** is started by executing an init command on the target device.

The target device can be connected using different solutions:

- How to connect over USB (default)
- How to connect over Ethernet
- How to connect over Wi-Fi

As soon as it is connected, you can start testing your application with `Android Studio`.

It is also possible to execute all ADB commands (see ADB commands for more details).

For that purpose, search for the ADB tool in your environment: Host computer installation.

Information

If you are using a Distribution Package, please refer to ADB for Distribution Package

2 Host computer installation

ADB is part of the Android SDK (loaded using `Android Studio`) for Linux®, Windows® and OS X®.

At this stage, ADB commands can be executed directly from the `Android Studio` terminal:

```
cd <SDK path>
cd platform-tools
```

The *<SDK path>* can be determined by opening the SDK Manager from `Android Studio`.

3 Commands

The ADB commands enable the execution of a variety of actions on the target device, such as installing and debugging apps. They also provide access to a Unix shell that can be used to run a variety of commands.

See ADB commands summary for more information.

Several useful commands:

- `adb devices`: check connected devices
- `adb logcat`: trace
- `adb root`: give root privileges
- `adb shell`: open a console on the device
- `adb remount`: remount read-only partition in read-write to allow updating their containing (use overlayfs mechanism)
- `adb push <file>`: push a file on the device
- `adb pull <file>`: pull a file from the device
- `adb install <package>`: install an application package (APK) on the device

4 Device connection

4.1 How to connect over USB

Follow the steps below to connect your device:

- Configure your host computer
- Configure your target device
- Check the device connection

4.1.1 Host computer configuration

First check if you can already see the device connected: Template:PC adb devices

The installation and configuration depend on the host computer OS used:

- Linux Host (Ubuntu) case
- Windows Host case
- OS X Host case

4.1.1.1 Linux® Host (Ubuntu)

In Linux environments, the Android USB drivers are built-in. The only action required is to set the target device information.

Information

To peform this action, you need administrator rights. In addition, you may need to add *sudo* in front of each executed command

To do this, open a terminal and

- Create (or update if it already exists) the *51-android.rules* file in */etc/udev/rules.d/* with the following information
  - *idVendor* = 0483 (STMicroelectronics vendor)
  - *idProduct* = 0adb (ADB on STMicroelectronics device)
  - *Mode* = 0666 (read/write permissions)
  - *Group* = plugdev (Unix group which owns the device node)

Information

Check if you belong to the *plugdev* group by executing the following command:

```
$ groups
```

Example:

```
# ADB on STMicroelectronics devices
SUBSYSTEM=="usb", ATTR{idVendor}=="0483", ATTR{idProduct}=="0adb", MODE="0660", GROUP="
plugdev"
```

- Make sure that the access rights to the created file are the correct ones:

```
chmod a+r /etc/udev/rules.d/51-android.rules
```

At this stage, your USB driver is correctly installed and configured.

4.1.1.2 *Windows Host*

In Windows environments, it is required to install the USB driver and set the target device information.

Get back a compatible driver here after (Windows 10): st-android-winusb.zip

Install it:

- Connect the device to your computer's USB port.
- Open Windows **Settings**, then select **Devices**, then '"Devices and Printers"'
- Right-click the name of the device you connected, and then select **Properties**
- In the **Hardware** tab, select again **Properties**
- In the **Driver** tab, you can select **Update Driver** and then select **Browse my computer for driver software** and click **Next**
- Locate the USB driver folder you just loaded
- Click **Next** to install the driver.

At this stage, the USB driver is correctly installed and configured.

4.1.1.3 *OS X Host*

In OS X environments, the Android USB drivers are built-in and no manual configuration is required to install and configure the USB driver.

4.1.2 Target device configuration

Prerequisite: Use ST Android distribution to start the target device and access the Android user interface.

On the target device:

- Disconnect the USB cable between the target device and the host computer (if it is connected).
- Find the *Settings > Developer options* configuration screen on your target device (if it is not visible, select *Settings > About device* and click the *Build number* menu entry seven times).
- Enable the USB Debugging option from the *Settings > Developer options* menu.
- Reconnect the USB cable between the target device and the host computer.
- If an alert is displayed on your target device requesting permission to "Allow USB debugging," click *OK. It is recommended to check the "Always allow from this computer" option to avoid this from happening each time the target device is connected to the same host computer.*

4.1.3 Checking the device connection

First, ensure that the device is connected on the host computer through an USB cable, and that it has been started.

With `Android Studio`, it's easy to check if the target device is available. For example, click on `Logcat` at the bottom and check the devices seen in the list, `No connected devices` is seen otherwise.

Information

The name of the STM32MPU devices start with *STMicroelectronics*

4.2 How to connect over Ethernet

To use ADB over an Ethernet connection, connect the host computer and the target device on the same network.

First, follow the instructions provided in How to connect over USB.

Then:

- Connect the USB cable between the target device and the host computer.
- Open a terminal on your host computer.
  - Go to the *<SDK path>/platform-tools/* directory (see Host computer installation to know how to get the <SDK path>).
  - Execute *adb shell ifconfig* to get the target IP address <device_ip_address>.
  - Execute *adb tcpip 5555* to set the TCP/IP connection over port 5555.
- Disconnect the USB cable between the target device and the host computer.
- Execute *adb connect <device_ip_address>* to reconnect on the device through Ethernet

4.3 How to connect over Wi-Fi

Follow the instructions provided in the *Android Studio* website: Connect to a device over Wi-Fi

5 ADB for Distribution Package

5.1 Installing the Host computer

By default, *ADB* is built when you compile the ST Android distribution. It is automatically added to your environment's $PATH.

Another possibility is to install ADB but executing *apt-get install android-tools-adb*, although this may be a hindrance to the development in case of ST Android distribution and OS ADB version mismatch (the server is restarted every time it is used with a mismatched version of an ADB client).

The device is not necessarily visible in the terminal when executing `adb devices`. If this is not the case:

- Create a file within ~/.android/ directory named adb_usb.ini.
- Add one line containing the STMicroelectronics idVendor:

```
0x0483
```

5.2 Implementing ADB

ADB source code is stored in *system/core/adb* (common files between host and target device are differentiated by the ADB _HOST tag).

Three properties can be configured for ADB:

- `persist.adb.trace_mask` used to enable or disable ADBD traces on target device.

  Possible values (*listed in* `system/core/adb/adb_trace.cpp`): 0, 1 (or all), adb, sockets, packets, rwx, usb, sync, sysdeps, transport, jdwp, services, auth, fdevent or shell

```
Board $> stop adbd
Board $> setptop persist.adb.trace_mask = <value>
Board $> start adbd
```

Information

Traces are available in `/data/adb/adb-<date_at_start>`

- `ro.adb.secure` used to enable or disable the authentication mechanism (1: authentication enabled, 0: authentication disabled). This property can not be modified dynamically. It is considered only in case of `eng` or `userdebug` image builds. It is ignored otherwise (the authentication mechanism is enabled by default for user build).
- `service.adb.tcp.port` used to enable the TCP/IP port if a wireless connection is used (this property is not set by default as it is not necessary for USB).

ADB is an Android specific gadget device available in user space. functionfs must be mounted since it is used to create this gadget device and register it to the USB Device Controller (UDC). Three endpoints are used (one for control, one for data input, one for data output).

It is also possible to create a composite device between ADB and other protocols, using the configfs mechanism.

ADB over USB is the default configuration within the Android baseline. By default, it is configured as follows:

- `persist.sys.usb.config`: [adb] USB ADB gadget device
- `sys.usb.config`: [adb] USB ADB gadget device
- `service.adb.root`: [0] no root privileges

The ADB over wireless connection can be set within your distribution (Ethernet or Wi-Fi) by adding the following property in */device/stm/<STM32Series>/<BoardId>/device.mk*:

```
PRODUCT_PROPERTY_OVERRIDES += service.adb.tcp.port=5555
```

Android debug bridge (Android specific)

Android debug bridge daemon (Android specific)

Software development kit (A programming package that enables a programmer to develop applications for a specific platform.)

Linux® is a registered trademark of Linus Torvalds.

Operating System

technology for wireless local area networking with devices based on the IEEE 802.11 standards

Read Only

USB Device Controller

Configuration File System (See https://en.wikipedia.org/wiki/Configfs for more details)

stm32mp1

eval disco (Generic term used, to complete configuration modules paths depending on used board)

Stable: 26.09.2019 - 14:30 / Revision: 26.09.2019 - 12:48

A quality version of this page, approved on *26 September 2019*, was based off this revision.

Overview

Git is a version control system (VCS) that is used for software development and other version control tasks. As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

Git was created by Linus Torvalds in 2005 for the development of the Linux® kernel, with other kernel developers contributing to its initial development.

As with most other distributed version control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version tracking abilities, independent of network access or a central server.

Like the Linux® kernel, Git is free software distributed under the terms of the GNU General Public License version 2.

- Introduction to Git

Description: Join kernel maintainer James Bottomley as he introduces you to Git, the version control system designed by and for the Linux® kernel development community. Learn about the history of Git and some of the basic functionality, including a brief demo.

- Git Getting Started and Reference Manual

Description: Website of Git community, containing a Getting Started document, and the full Git reference manual.

- Git Documentation list provided by Kernel.org

Description: Kernel.org WIKI page for Git online documents.

Stable: 25.03.2021 - 16:36 / Revision: 25.03.2021 - 16:36

A quality version of this page, approved on *25 March 2021*, was based off this revision.

Information

Recommended setup: Native Linux® PC

Contents

1 Purpose

This article explains and describes the host PC hardware and software configuration required to activate and run the STM32 MPU platforms.

2 Recommended PC configurations

The PC requirements depend on the Package you want to use.

The table below guides through the selection and configuration of the host PC environment according the targeted Package:

Host environment	Starter Package	Developer Package	Distribution Package
Windows® (64 bits) Tested with Windows 7 and Windows 10 Preferred version is Windows 10	native	Virtual machine	Virtual Machine
Linux (64 bits) Tested with Ubuntu®**18.04** and 20.04	Native	Native + additional packages (see Linux PC chapter)	native + additional packages (see Linux PC chapter)

There are no absolute minimal requirements regarding the PC hardware configuration. However ST recommends to meet or exceed the following hardware requirements when using the *Developer Package* or *Distribution Package*.

The table below corresponds to the minimal validated configuration:

Hardware item	Minimal validated configuration	Comments / Recommendations
CPU	core i5-2540M @ 2.6 GHz 2 cores (4 threads) 3-Mbyte cache	**64-bit instruction set is mandatory** **8 cores/threads or more** is a good configuration for *Developer Package* and *Distribution Package*.
RAM	8 Gbytes	**16 Gbytes or more** are recommended especially for *Virtual machine* setup, *Developer Package* and *Distribution Package*.
Hard drive	320 Gbytes	**1 Tbytes** is the best suited configuration for *Distribution Package*

3 Linux® PC

A Linux PC with **Ubuntu 18.04** is the recommended setup. Other Ubuntu revisions are also supported (refer to Yocto reference manual[1]) .

Information

ecosystem release v2.1.0

ST solutions are tested and validated on a Linux PC running Ubuntu 18.04 LTS (and Ubuntu 20.04 LTS).

Information

ecosystem release v2.0.0

ST solutions are tested and validated on a Linux PC running Ubuntu 18.04 LTS (and Ubuntu 16.04 LTS).

3.1 Checking Internet access

- Internet access through http and https protocols

Required at least for *Developer Package* and *Distribution Package*.

The command below enables checking internet access through http/https protocols:

```
PC $> wget -q www.google.com && echo "Internet access over HTTP/HTTPS is OK !" || echo
"No internet access over HTTP/HTTPS ! You might need to set up a proxy."
```

If an 'OK' message is returned, the network is well configured. In such case, skip the rest of this section and jump to Install extra packages.

Any other situation likely indicates that a proxy is required for http/https protocols. The best solution to set a proxy for http/https protocols is via the shell variables http_proxy and https_proxy:

```
PC $> export http_proxy=http://<MyProxyLogin>:<MyProxyPassword>@<MyProxyServerUrl>:<MyProx
yPort>
PC $> export https_proxy=http://<MyProxyLogin>:<MyProxyPassword>@<MyProxyServerUrl>:<MyPro
xyPort>
```

Since your password (<MyProxyPassword>) might contain "special characters", translate it into ASCII hexacode. To do this, use the command below:

PC $> echo -n " <MyProxyPassword>" | od -A n -t x1 -w128 | head -1 | tr " " "%"

Check again the Internet access by using the following command:

```
PC $> wget -q www.google.com && echo "Internet access over HTTP/HTTPS is OK !" || echo
"No internet access over HTTP/HTTPS ! You might need to set up a proxy."
```

- **Internet access for *sudo* commands**

Required for *Distribution Package*.

By default, *sudo* commands are executed in the *root* user environment; no Internet proxy settings are applied for *root* users. *Root* users must be able to browse Internet after having created an alias passing the proxy settings on *sudo* commands:

```
PC $> alias sudo='sudo http_proxy=$http_proxy'
```

Check that the *sudo* commands are successful (requires Internet access):

```
PC $> sudo apt-get update
```

- **Internet over git://, ssh:// and others specifics protocols**

Required for *Distribution Package*.
In addition to http/https protocols (used in 90% of the Internet traffic), other protocols such as git:// or ssh:// might be required.

For example, in the context of the *Distribution Package*, some "git fetch" commands might require "git:// protocols".
To support these protocols through a proxy, directly setup the proxy in the $HOME/.gitconfig file (core.gitproxy) and use a tool such as *cockscrew*[2] to tunnel the git:// flow into the http flow:

```
PC $> sudo apt-get update
PC $> sudo apt-get install corkscrew

PC $> git config --replace-all --global core.gitproxy "$HOME/bin/git-proxy.sh"
PC $> git config --add --global core.gitproxy "none for <MyPrivateNetworkDomain>"
(optional, for example .st.com or localhost)
PC $> echo 'exec corkscrew <MyProxyServerUrl> <MyProxyPort> $* $HOME/.git-proxy.auth' >
$HOME/bin/git-proxy.sh
PC $> chmod 700 $HOME/bin/git-proxy.sh
PC $> echo '<MyProxyLogin>:<MyProxyPassword>' > $HOME/.git-proxy.auth
PC $> chmod 600 $HOME/.git-proxy.auth
```

Use the command below to test the proxy settings:

```
PC $> git ls-remote git://git.openembedded.org/openembedded-core > /dev/null && echo OK
|| echo KO
```

The command returns 'OK' if the proxy settings are correct.

3.2 Installing extra packages

Required for *Developer Package* and *Distribution Package*.

Additional Ubuntu packages must be installed to perform basic development tasks, basic cross-compilation (via *Developer Package*) or more complex cross-compilation such as OpenEmbedded does (via *Distribution Package*):

- Packages required by OpenEmbedded/Yocto (details here):

```
PC $> sudo apt-get update
PC $> sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-
essential chrpath socat cpio python3 python3-pip python3-pexpect xz-utils debianutils
iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3 pylint xterm
PC $> sudo apt-get install make xsltproc docbook-utils fop dblatex xmlto
```

- Packages needed for some "Developer Package" use cases:

```
PC $> sudo apt-get install libncurses5 libncurses5-dev libncursesw5-dev libssl-dev linux-
headers-generic u-boot-tools device-tree-compiler bison flex g++ libyaml-dev
```

- Package for repo (used to download the "Distribution Package" source code):

```
Please follow the installation instructions described in repo.
 For Ubuntu 16.04, use the legacy repo old-repo-python2 chapter
```

- Useful tools:

```
PC $> sudo apt-get install coreutils bsdmainutils sed curl bc lrzsz corkscrew cvs
subversion mercurial nfs-common nfs-kernel-server libarchive-zip-perl dos2unix texi2html
diffstat libxml2-utils
```

You can also install a Java Runtime Engine that is required for STM32CubeMX and STM32CubeProgrammer

```
PC $> sudo apt-get install default-jre
```

3.2.1 Installing extra packages for Android™

Below information is related to the Android™ distribution

Before downloading and building the STM32MPU distribution for Android™, make sure your system meets the following requirements:

- A 64-bit Linux® environment.
- At least 150 Gbytes of free disk space. If you conduct multiple builds, even more space is required.
- At least 8 Gbytes of RAM/swap. If you are running Linux on a virtual machine, at least 16 Gbytes of RAM/swap are required.

For more details, refer to the requirements page of the AOSP website.

Use the following commands to install the packages required to build an environment for Android (Distribution Package), after having installed the required packages listed on Android website:

```
sudo apt-get update
sudo apt-get install chrpath curl libxml2-utils gdisk pv python-
pycryptopp python-crypto autotools-dev automake libusb-1.0-0-dev
```

To ensure USB communication (with ADB) between the host and the device, see ADB § Device Connection.

To run Android-provided tests (CTS and VTS), see Android Platform Testing.

At this stage: Your environment is ready for Android build, debug and test.

3.3 Additional configurations

- Allowing up to 16 partitions per MMC

By default a maximum of 8 MMC partitions are allowed on Linux systems. All Packages (such as Starter Package) need more than 10 partitons for the storage device. To extend the number of partitions per device to 16, the following options must be added to modprobe:

```
PC $> echo 'options mmc_block perdev_minors=16' > /tmp/mmc_block.conf
PC $> sudo mv /tmp/mmc_block.conf /etc/modprobe.d/mmc_block.conf
```

- Checking for *locale* setup

Required for *Distribution Package*.

The *locale* setting is used by some applications/commands (including by *Distribution Package* applications/commands).
Verify that the *locale* settings are as follows:

```
PC $> locale
LANG=en-US.UTF-8
```

If the *locale* command returns a different configuration than the one shown above, reconfigure it as follows:

```
PC $> sudo update-locale LANG=en_US.UTF-8
```

- Adding users in basics groups

The *user* login must belong to the basic Linux groups such as **disk**, **tty**, **dialout** or **plugdev**
Use the *groups* command to list groups for the current user:

```
PC $> groups
```

If needed, add *user* to the missing *<groups>*:

```
PC $> sudo adduser $USER <group>
```

Then **reboot** the PC.

3.4 Seting up *Git* user information

Required for *Developer Package* and *Distribution Package*.

The user Information are needed by git[3] if *commit* and/or *push* commands are used:

```
PC $> git config --global user.name "Your Name"
PC $> git config --global user.email "you@example.com"
```

4 Windows PC

Starter Package **may run on Windows.**
Developer Package **and** ***Distribution Package*** **require a Linux environment.**

Warning

ST solutions, while reportedly functional when running on a Linux Virtual machine, are only validated for Linux native setups.

There are several ways to run Linux system on top of a Windows host PC. ST recommends to use a Virtual Machine System:

1. Install a virtual machine such as VMWare [4]
2. Setup a **64-bit** Ubuntu image compatible with your virtual machine

ST, in an experimental way, has also run *Developer Package* and *Distribution Package* on WSL2 (Windows Subsystem for Linux 2). Refer to WSL2 chapter.

4.1 Virtual machine system

4.1.1 Installing the virtual machine

ST has selected VMWare as Linux virtual machine solution.

VMWare is a commercial company specialized in virtualization solutions. The available solutions to support a virtual Linux machine on a Windows PC are:

- VMWare Workstation Player (paid solution) for commercial use (download here [5])
- VMWare Workstation Player (free solution) for home use (download here [6])

Please proceed with the installation of the virtual machine.

Before running the virtual machine, make sure the virtualization is activated in the BIOS (it should be activated by default for any retail PC).

4.1.2 Downloading the Ubuntu image for the virtual machine

The "osboxes.org" [7] website provides virtual machine images compatible with VMWare(*.vmdk).

ecosystem release v2.1.0 : **Setup have been validated and tested on Ubuntu 18.04 (64bits)** and Ubuntu 20.04 (64bits).
ecosystem release v2.0.0 : **Setup have been validated and tested on Ubuntu 18.04 (64bits)** and Ubuntu 16.04 (64bits).

Download the 64-bit Ubuntu image available at [8], then:

1. Unzip the downloaded file
2. In VMware, create a virtual machine using the Ubuntu virtual disk downloaded from osboxes.org.

The recommended usage is to dedicate, at least, half of the host machine to the virtual machine:

```
- CPU: 2 cores at least,
- RAM: 6 Gbytes or more is a good choice (the more RAM allocated to Virtual Machine the
better - the RAM allocated to Virtual Machine must be at least 4 Gbytes),
- Network: NAT is a good easy way to benefit from a network connection within the virtual
machine.
```

The virtual size of the virtual disk downloaded from osboxes.org is about 500 Gbytes. Even if, at beginning, the real size of the file of the virtual disk is less, the size can grow up to 500 Gbytes over distribution package compiling or package development.

Information

For VMware, first create a default virtual machine, then add the previously downloaded *.vmdk* file.

Please refer to the VMwarePlayer screenshot tutorial.

4.1.3 Launching Ubuntu image

Warning

For "AZERTY" keyboard users:

The default keyboard configuration is "QWERTY".
To configure the keyboard to "AZERTY", start by opening a session (take care that the keyboard layout is QWERTY).
TIP: the password for the default user "*osboxes.org*" is "*osboxes.org*".
TIP: the '.' character is obtained by clicking ':' on an AZERTY keyboard configured in QWERTY.
Once the session is open, click the 'En' icon on top/right of the screen, select the French ('Fr') keyboard layout and move it to the first position in the list.
Optionally the 'En' keyboard can be completely removed. If the 'Fr' option is not present, it can be added with the 'Text entry setting' menu.

Default 'Ubuntu *Credentials'* **are set to "osboxes.org" for both login and password.**

Warning

Adjusting screen resolution:

The (default) resolution used by the virtual machine is 800x600 (smallest available). It is not automatically adjusted to the display resolution. To adjust the resolution, click the "settings" icon ('toothed wheel' on top/right of the screen), then "system settings ..." > "display" and select the appropriate resolution for the display (do not to forget to click the "Apply" button on bottom/left of the "Screen Resolution Setting" window).

For a better experience with the VMware virtual machine, install "vmware-tools" in order to be able to use the clipboard to drag-and-drop and copy/paste files between VMware and Windows. A step-by-step installation procedure of **vmware-tools** is available in PreRequisite-Vmware-tools.pdf.

The virtual machine is now up and running!

The Ubuntu setup must be finalized according recommendations provided in Linux PC chapter

Warning

USB connection speed:

A USB connection is required to access STLink (debugger and serial port) and by STM32CubeProgrammer. The speed of the USB connection between Linux running in the virtual machine and the external USB devices can be severely impacted by the following factors:

- the virtual machine USB setup
- the USB controller in the host PC
- the USB device connected to host PC
- any USB hub between the USB host and the USB device.

If the speed of your USB connection is too low:

- try different USB configurations of the virtual machine;
- connect the USB device directly to the host USB port (without any USB hub);
- try connecting the USB device to another USB port of the host (some PC have different USB controller on different USB port).

4.2 WSL2 (experimental)

Even if STMicroelectronics strongly recommends to use a Linux® environment, the *Developer Package* and *Distribution Package* work on WSL2 (Windows Sub-system Linux 2) environment. WSL is a feature provided by **Windows 10®**.
WSL2 is a new version of the architecture that powers the Windows subsystem for Linux to run ELF64 Linux binaries on Windows (more details on aka.ms/wsl2). It is available on Windows 10 since build 18917. The *Developer Package* successfully runs on WSL2 but unsuccessfully on WSL.

- Installing WSL2 :
  - To install WSL2, read https://docs.microsoft.com/fr-fr/windows/wsl/wsl2-install
  - Once WSL2 is installed, jump to #Linux_PC to make your WSL2 ready to run *Developer Package* and/or *Distribution Package*.
- WSL2 limitations :
  - Up to now (September 2019), WSL2 does not support hardware such as USB or serial devices (more details).
    This means that STM32CubeProgrammer must be used through native Windows.
  - WSL2 files are not browsable from Windows native file explorer.
    To share files between WSL2 and Windows, it is recommend to use the mount point */mnt/c* from WSL2 and perform copies.
- Tips for using WSL2 :
  - Launching graphical application : on *wiki.ubuntu.com*, go to WSL page and read the chapter Running Graphical Applications.

4.3 References

- Supported Linux Distributionsl
- https://en.wikipedia.org/wiki/Corkscrew_(program)
- Git
- http://vmware.com
- https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/15_0
- https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html
- http://osboxes.org
- https://www.osboxes.org/ubuntu/#ubuntu-1804-vmware

Linux® is a registered trademark of Linus Torvalds.

Microprocessor Unit

Central processing unit

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today for volatile random-acces ssemiconductor memories.)

Android Open Source Project

Android debug bridge (Android specific)

Compatibility Test Suite (Android specific) or Clear to send (in UART context)

Vendor Test Suite (Android specific)

MultimediaCard

ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families (See ST-LINK for more details)

Stable: 20.12.2019 - 19:12 / Revision: 12.12.2019 - 12:53

A quality version of this page, approved on *20 December 2019*, was based off this revision.

This article describes the ST-LINK hardware probes.

Contents

1 Introduction

The **ST-LINK**[1] is an in-circuit debugger and programmer for the STM8 and STM32 microcontroller families.

ST-LINK is a USB device and has to be connected to a PC host. It can be either embedded on ST boards or provided as standalone dongle.

ST-LINK can support different debug protocols depending on ST-LINK hardware version and on its embedded firmware version:

- SWIM: debug protocol for STM8 microcontrollers
- SWD/JTAG: debug protocol for STM32 microcontrollers and microprocessors

and communication interfaces:

- UART
- I2C
- SPI
- CAN
- GPIO

2 Hardware versions

Several versions of ST-LINK exist: ST-LINK/V1[1], ST-LINK/V2[2], ST-LINK/V2-A, ST-LINK/V2-B, ST-LINK/V2-1 and STLINK-V3SET[3].

For details about the different versions, please refer to technical note[4] of ST-LINK derivatives.

To find out the ST-LINK version that is embedded in your ST board, refer to the Category:STM32 MPU boards page, and then select your hardware board.

3 Getting started

3.1 Installing the USB driver

Two USB drivers are associated to ST-LINK, depending of ST-LINK version: one for the debugger itself, and one for the serial communication port from (***ST-LINK/V2-1***).
The serial communication port uses standard CDC ACM USB Class, which is usually present by default on all PC operating systems. The USB driver for the debugger can differ depending of the PC operating system:

- **MS Windows®**

A driver must be installed before connecting ST-LINK to a Windows® 7, Windows® 7 8, or Windows® 10 PC via the USB.

The driver is automatically installed by the toolsets supporting ST-LINK. It is also available from www.st.com [5].

- **Linux®**

Users must be granted with rights for accessing ST-Link USB devices. Rules must then be added into /etc/udev/rules.d.

All information and files for installing the udev rules are provided in the **STSW-LINK007**[6] package available from www.st.com (see *stsw-link007\AllPlatforms\StlinkRulesFilesForLinux\Readme.txt* file).

- **Mac OS®**

No specific installation is required.

3.2 Connecting JTAG/SWD for debugging

- Embedded ST-LINK

A JTAG/SWD link is available from the USB link provided by the ST-LINK. The USB device is mounted on the host PC and ready to be used.

- Standalone ST-LINK

Pins are available on the ST-LINK to connect the JTAG/SWD signals. Refer to Hardware versions for connection details.

- **JTAG**: VCC, JTDI, JTMS, JCLK, JRCLK, JTDO, NRST and GDN signals must be connected to the JTAG/SWD connector.
- **SWD**: VCC, SWCLK, SWDIO, NRST, SWO and GND signals must be connected to the JTAG/SWD connector (on some ST-LINK hardware version, a dedicated SWD port can also be available).

3.3 Connecting UART port (from *ST-LINK/V2-1*)

- Embedded ST-LINK

A UART serial port is available from the USB link provided by the ST-LINK. The USB device is mounted on the host PC and ready to be used.

- Standalone ST-LINK

Pins are available on the ST-LINK to connect the Rx/Tx and GND signals. Refer to Hardware versions for connection details.

4 To go further

4.1 Updating the embedded firmware

All information are given in the STSW-LINK007[6] software package available from www.st.com.

Warning

The firmware update application embeds the latest firmware version, so do not hesitate to get the latest version of the executable STSW-LINK007[6] software package

4.2 How to bypass the embedded ST-LINK

To use a newest version of ST-LINK standalone probe or another debug probe, ST-LINK can be bypassed for ST boards that already embed it.

- Deactivating the embedded ST-LINK

Please refer to the board description and schematic for how to put the embedded ST-LINK in reset mode if available.

- Bypassing the embedded ST-LINK signals and connecting an external hardware probe

Connect the relevant signals depending on the interface available on the new hardware probe. Please refer to Connecting JTAG /SWD for debug and Connecting UART port for standalone external ST-LINK.

5 References

- 1.01.1 https://www.st.com/en/development-tools/st-link.html
- https://www.st.com/en/development-tools/st-link-v2.html
- https://www.st.com/en/development-tools/stlink-v3set.html
- https://www.st.com/resource/en/technical_note/dm00290229.pdf
- https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-utilities/stsw-link009.html
- 6.06.16.2 https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/stsw-link007.html

Single Wire Interface Module (debug protocol for STM8 microcontrollers)

Serial Wire Debug

debug and test protocol, named from the Joint Test Action Group that developed it

Universal Asynchronous Receiver/Transmitter

Inter-Integrated Circuit (Bi-directional 2-wire bus standard for efficient inter-IC control.)

Serial Peripheral Interface

Controller Area Network (robust bus mainly used for automotive applications)

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Linux® is a registered trademark of Linus Torvalds.

spelling for older versions of STLink, ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families

Operating System

Stable: 23.09.2020 - 13:22 / Revision: 12.06.2020 - 13:25

A quality version of this page, approved on *23 September 2020*, was based off this revision.

1 STM32CubeMX overview

This article describes STM32CubeMX, an official STMicroelectronics graphical software configuration tool.

The STM32CubeMX application helps developers to use the STM32 by means of a user interface, and guides the user through to the initial configuration of a firmware project.

It provides the means to:

- configure pin assignments, the clock tree, or internal peripherals
- simulate the power consumption of the resulting project
- configure and tune DDR parameters
- generate HAL initialization code for Cortex-M4
- generate the Device Tree for a Linux kernel, TF-A and U-Boot firmware for Cortex-A7

It uses a rich library of data from the STM32 microcontroller portfolio.

The application is intended to ease the initial development phase by helping developers to select the best product in terms of features and power.

2 STM32CubeMX main features

- Peripheral and middleware parameters

  Presents options specific to each supported software component

- Peripheral assignment to processors

  Allows assignment of each peripheral to Cortex-A Secure, Cortex-A Non-Secure, or Cortex-M processors

- Power consumption calculator

  Uses a database of typical values to estimate power consumption, DMIPS, and battery life

- Code generation

  Makes code regeneration possible, while keeping user code intact

- Pinout configuration

  Enables peripherals to be chosen for use, and assigns GPIO and alternate functions to pins

- Clock tree initialization

  Chooses the oscillator and sets the PLL and clock dividers

- DDR tunning tool

  Ensures the configuration, testing, and tuning of the MPU DDR parameters. Using U-Boot-SPL Embedded Software.

3 How to get STM32CubeMX

Please, refer to the following link STM32CubeMX to find STM32CubeMX, the Release Note, the User Manual and the product specification.

Doubledata rate (memory domain)

Hardware Abstraction Layer

Cortex®

Linux® is a registered trademark of Linus Torvalds.

Trusted Firmware for Arm Cortex-A

Das U-Boot -- the Universal Boot Loader (see U-Boot_overview)

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Microprocessor Unit

Stable: 23.03.2021 - 10:30 / Revision: 23.03.2021 - 10:09

A quality version of this page, approved on *23 March 2021*, was based off this revision.

This article gives a short introduction to the official STM32CubeProgrammer tool: What are the flashing principles? Where to download the software? How to install it? How to use it?

Contents

1 STM32CubeProgrammer overview

STM32CubeProgrammer is the official STMicroelectronics tool for creating partitions into any Flash device available on STM32 platforms.
Once created, STM32CubeProgrammer allows populating and updating the partitions with the prebuilt binaries.

The connection between the host PC and the board can be done through UART or USB serial links.

Any Flash device supported on STM32MPU boards can be flashed using STM32CubeProgrammer. For example, for the STM32MP157 Evaluation board:

1. microSD™ card
2. eMMC
3. NAND Flash memory
4. NOR Flash memory

Information

During development, it is recommended to use a microSD card; you can then switch to eMMC or NAND Flash memory.

More details on supported Flash memory technologies and mapping can be found in Flash mapping articles, e.g. STM32MP15 Flash mapping.

2 STM32CubeProgrammer installation

2.1 Installing the STM32CubeProgrammer tool

<table>
<tr><th></th><th>STM32CubeProgrammer for Linux® host PC</th><th>STM32CubeProgrammer for Windows® host PC</th></tr>
<tr><td>Download</td><td colspan="2">Version 2.7.0
• Browse SetupSTM32CubeProgrammer link and follow instructions to download the package, you need a myST account.
• Download the archive file on your host PC in a temporary directory
• Uncompress the archive file to get the STM32CubeProgrammer installers:

PC $>unzip en.stm32cubeprog.zip</td></tr>
<tr><td>Installation</td><td>• Execute the Linux installer, which guides you through the installation process.

<pre>$> ./SetupSTM32CubeProgrammer-2.7.0.linux</pre>
• The path to the STM32CubeProgrammer binary must be added to the PATH environment variable
  • either in each Terminal program in which the STM32CubeProgrammer binary needs to be used, using the following command:</td><td>• Execute the Windows installer, which guides you through the installation process.</td></tr>
</table>

	STM32CubeProgrammer for Linux® host PC	STM32CubeProgrammer for Windows® host PC

	`$> export PATH=<my STM32CubeProgrammer install directory>/bin:$PATH`
	• or once for all by creating a link to the STM32CubeProgrammer binary in a directory already present in PATH. For example, if "/home/bin" is in the PATH environment variable, run the following command:
	`$> ln -s <my STM32CubeProgrammer install directory>/bin/STM32_Programmer_CLI /home/bin/STM32_Programmer_CLI`	

User manual	• Instructions to follow for using the STM32CubeProgrammer can be found in user manual, UM2237 available from ST web site, or in STM32CubeProgrammer#How to flash with STM32CubeProgrammer article.	
Detailed release note	• Details about the content of this tool version are available from ST web site at Release Note .	

2.2 Preparing the USB serial link for flashing

It is recommended to use the USB (in DFU mode) for flashing rather than the UART, which is too slow.

Below indications on how to install the USB in DFU mode under Linux and Windows OS, respectively.

- **For Linux host PC** or Windows host PC with VMWare:

The libusb1.0 package (including USB DFU mode) must be installed to be able to connect to the board via the USB port. This is achieved by typing the following command from the host PC terminal:

```
PC $> sudo apt-get install libusb-1.0-0
```

To allow STM32CubeProgrammer to access the USB port through low-level commands, proceed as follows:

```
PC $> cd <your STM32CubeProgrammer install directory>/Drivers/rules
PC $> sudo cp *.* /etc/udev/rules.d/
```

- **For Windows host PC**:

Run the “STM32 Bootloader.bat” file to install the STM32CubeProgrammer DFU driver and activate the STM32 microprocessor device in USB DFU mode. This driver (installed by STM32 Bootloader.bat) is provided within the STM32CubeProgrammer release package. It is located in the DFU driver folder, \Drivers\DFU_Driver.

In case of issue, refer to How to proceed when the DFU driver installation fails on Windows host PC.

To validate the installation, the DFU driver functionality can be verified by following the FAQ instructions provided in how to check if the DFU driver is functional.

STM32CubeProgrammer is now ready to use: to know more about this tool, please continue reading; otherwise, jump to the Starter or Distribution Package article corresponding to your board: Category:Starter Package or Category:Distribution Package.

3 Flash programming principles

Flash programming consists in transferring the binaries stored on the host computer into the platform Flash memory(ies), via a serial interface.
This operation requires a communication interface between the **STM32CubeProgrammer** and an **embedded programming service**.

The selection of the binaries to download and the Flash memory destination is done through the flashlayout.tsv file.
The STM32Cubeprogrammer tool uses the Flashlayout.tsv file as an input.
The Flashlayout includes a formal description of the partitions (ID, naming, type, offset) as well as the identification of the Flash memory to be populated.

Some default "tsv" files aligned with the STM32 Flash memory mapping (e.g.STM32MP15_Flash_mapping) are provided in the STM32Cubeprogrammer tool. They can be used as a starting point and can be further adapted to specific application needs.

More examples as well as information on the description format are available in STM32CubeProgrammer flashlayout.

See AN5275: USB DFU/USART protocols used in STM32MP1 Series bootloaders for protocol details.

4 How to flash with STM32CubeProgrammer

Once the STM32CubeProgrammer has been installed, complete flashing procedures for STM32 boards are available in Starter Package and Distribution Package articles. Select the article corresponding to your STMicroelectronics platform.

The generic syntax of the STM32CubeProgrammer command is explained below.

- **With Linux host PC**, the main command is:

```
PC $> ./bin/STM32_Programmer_CLI  + options given below
```

- **With Windows host PC**, the main command is:

```
PC $> ./bin/STM32_Programmer_CLI.exe + options given below
```

The Linux related information provided below can be extrapolated for other host PC operating systems:

- **Complete flashing command (for Linux host PC):**

```
PC $>  STM32_Programmer_CLI -c port=<DEVICE_PORT_LOCATION> -w [<file.tsv>]
where
 w:
     Write
 <file.tsv>:PathToFlashlayout/flashlayout.tsv file (see above)
         if Flashlayout and binaries are not in the same directory, then the path to the
Flashlayout files must be precised.
 <DEVICE_PORT_LOCATION>:
         e.g. usb1 (case sensitive): for other options, please refer to How to find the
DEVICE_PORT_LOCATION parameter value for the USB link.
```

Populating all partitions can take a few minutes (depending mainly on the rootfs size).

5 FAQs

5.1 Where to find the STM32CubeProgrammer user manual

The user manual is available in ST deliveries under <STM32CubeProgrammer installation directory>/doc/UM2336.pdf (see Installing the STM32CubeProgrammer tool).

5.2 How to check if the DFU driver is functional

Prerequisites: STM32CubeProgrammer installation completed and STM32 board connected to the host PC.

First, activate the USB driver on Ubuntu on VMware Workstation Player (use VM Player menu "removable devices" to connect to STMicroelectronics DFU in HS mode), then execute the following command and check the result:

```
PC $> STM32_Programmer_CLI -l usb
```

- If the DFU is functional, the result is similar to the following

```
    -------------------------------------------------------------------
                  STM32CubeProgrammer <tool version>
    -------------------------------------------------------------------

Total number of available STM32 device in DFU mode: 1

 Device Index           : USB1
 USB Bus Number         : 002
 USB Address Number     : 008
 Product ID             : USB download gadget@Device ID /0x500, @Revision ID /0x0000
 Serial number          : 0000000000
 Firmware version       : 0x0110
 Device ID              : 0x0500
```

- Otherwise, the DFU is not functional and the result is the following:

```
-------------------------------------------------------------------
                  STM32CubeProgrammer <tool version>
-------------------------------------------------------------------


 Warning: No STM32 device in DFU mode connected
```

5.3 How to proceed when the DFU driver installation fails on Windows host PC

Prerequisites: STM32CubeProgrammer installation completed.

The STM32 Bootloader.bat allows to install the DFU driver on a Windows PC (see Preparing the USB serial link for flashing). In case of failure, here is a typical error log:

```
c:\demo\MPU\stm32_programmer_package_v1.0.3_MPU\Drivers\DFU_Driver>"STM32 Bootloader.bat"
c:\demo\MPU\stm32_programmer_package_v1.0.3_MPU\Drivers\DFU_Driver>echo off Microsoft PnP
Utility
Processing inf :            DFU_in_HS_Mode.inf
Failed to install the driver on any of the devices on the system : No more data is
available.
Total attempted:              1
Number successfully imported: 0
```

This error occurs either because the STM32CubeProgrammer DFU driver is already installed or because the DFUSE driver is installed.

To verify if the STM32CubeProgrammer DFU driver is correctly installed on your Windows host PC, please proceed as follow:

1. Connect the board to the Windows host PC.
2. Launch the Device Manager utility and execute the below action depending on your use case:
   - The STM32CubeProgrammer DFU driver is not installed:
     Execute the STM32 Bootloader.bat script to install it (see Preparing the USB serial link for flashing).

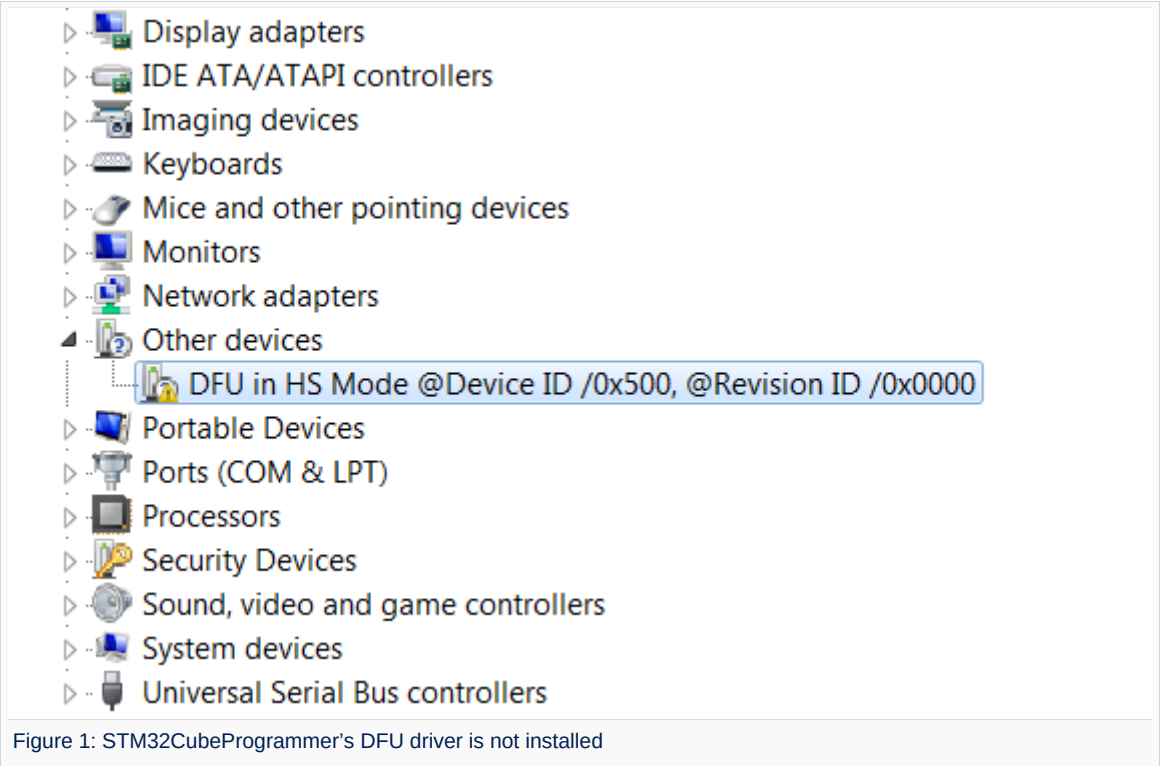


Figure 1: STM32CubeProgrammer's DFU driver is not installed

   - The DfuSe driver is installed and you must uninstall it.
     Right click **STM Device in DFU Mode** and select **Uninstall**.

Figure 1: STM32 DFU Device with DfuSe Driver

- The STM32CubeProgrammer DFU driver is installed and ready to be used.

Figure 2: STM32 DFU Device with STM32_Programmer Driver

5.4 How to find the DEVICE_PORT_LOCATION parameter value for the USB link

Prerequisites: STM32CubeProgrammer installation completed and STM32 board connected to the host PC.

To find the value of the DEVICE_PORT_LOCATION corresponding to the USB link, use the following command:

```
PC $> STM32_Programmer_CLI -l usb
     -------------------------------------------------------------------
                    STM32CubeProgrammer <tool version>
     -------------------------------------------------------------------

Total number of available STM32 device in DFU mode: 1

 Device Index           : USB1
 USB Bus Number         : 002
```

```
USB Address Number     : 008
Product ID             : USB download gadget@Device ID /0x500, @Revision ID /0x0000
Serial number          : 0000000000
Firmware version       : 0x0110
Device ID              : 0x0500
```

You can then report the value returned by the command line as explained in the How to flash with STM32CubeProgrammer chapter. Pay attention that the value must be written in **lower case**.

5.5 How to explore all the partitions of a populated microSD card

A populated microSD card can contain more than eight partitions, exceeding the number of partitions allowed by default in a Linux system.
At insertion step, you may get an 'Error mounting /dev/mmcblk0p9' or 'mount: cannot mount /dev/mmcblk0p9 read-only' error message.

By adding some options to modprobe, it is possible to allow more than eight partitions on a storage device:

```
PC $> echo 'options mmc_block perdev_minors=16' > /tmp/mmc_block.conf
PC $> sudo mv /tmp/mmc_block.conf /etc/modprobe.d/mmc_block.conf
```

5.6 How to update partitions

For partial Flash memory update, the flashlayout.tsv must be edited to select only the partition that needs to be updated (first column =1).
See STM32CubeProgrammer_flashlayout#Updating partitions for details and example, the rest of the procedure is the same.

On ST boards, if FSBL or SSBL partitions need to be updated, the Bin2boot fields must be configured with **ST binaries**, as described in STM32CubeProgrammer_flashlayout#Updating partitions using official ST bootloaders.

5.7 How to populate a microSD card inserted in a Linux host PC

STMicroelectronics distribution is delivered with the *create_sdcard_from_flashlayout.sh* script that allows to prepare and flash a microSD card image.

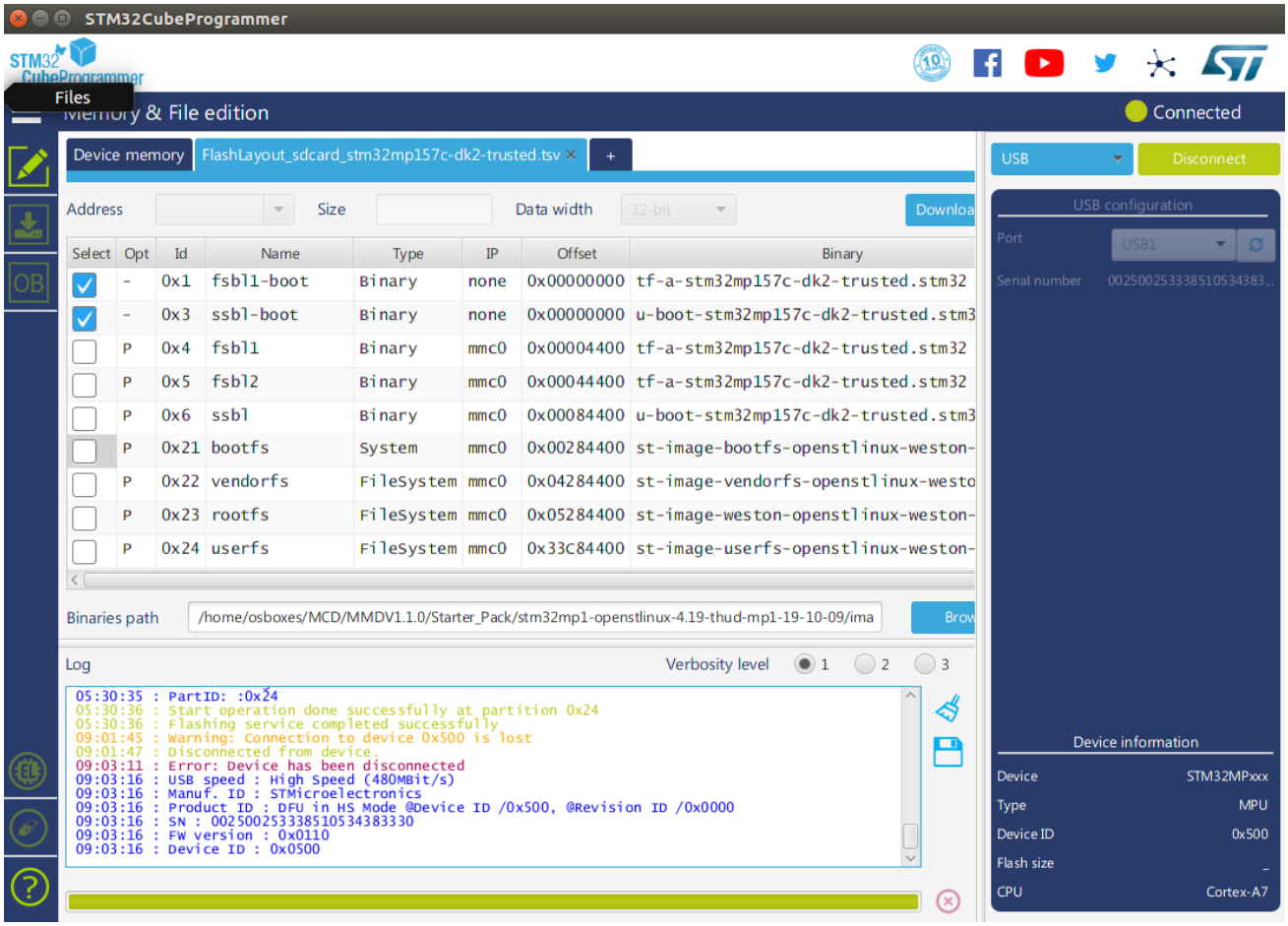
5.8 How to fuse STM32MP15x OTP

5.8.1 Connection

STM32CubeProgrammer requires to be connected to U-Boot in order to access OTP alternate.

To do it, you have to initiate a flash process but selecting only the 2 first "boot" partitions TF-A(0x1) and U-Boot(0x03) of a . tsv.

Using STM32CubeProgrammer GUI this can done without editing by using any available .tsv but selecting only the 2 first partitions thanks to "Select" column tick boxes on the left, as follow:



Perform the "Download" till success popup then disconnect from the GUI and switch to CLI in a terminal without rebooting the board:

```
PC $> STM32_Programmer_CLI -c port=usb1
```

5.8.2 Display status

It will also displayed the status for each OTP words (Shadow lock read/write, program stick lock, permanent write lock):

- All lower OTP
- Only upper that are available for non secure (defined in TF-A device tree).

To display the OTP:

```
PC $>  STM32_Programmer_CLI -c port=usb1 -otp displ

-------------------------------------------------------------------
                  STM32CubeProgrammer v2.3.0
-------------------------------------------------------------------
```

```
USB speed   : High Speed (480MBit/s)
Manuf. ID   : STMicroelectronics
Product ID  : USB download gadget@Device ID /0x500, @Revision ID /0x0000
SN          : 00250025333851053438333O
FW version  : 0x0110
Device ID   : 0x0500
Device name : STM32MPxxx
Device type : MPU
Device CPU  : Cortex-A7


UPLOADING OTP STRUCTURE ...
  Partition     : 242
  Size          : 1024 Bytes

Uploading OTP data:
[==================================================] 100%

OTP Partition read successfully


OTP STRUCTURE VERSION          :0x00000001

BSEC CONTROL REGISTERS:

BSEC_OTP_CONFIG                :0x0000000e
                                |_[08] TR[1]                    : 0
                                |_[07] TR[0]                    : 0
                                |_[06] PRGWIDTH[3]              : 0
                                |_[05] PRGWIDTH[2]              : 0
                                |_[04] PRGWIDTH[1]              : 0
                                |_[03] PRGWIDTH[0]              : 1
                                |_[02] FRC[1]                   : 1
                                |_[01] FRC[0]                   : 1
                                |_[00] PWRUP                    : 0

BSEC_OTP_Status                :0x00000041
                                |_[07] BIST2LOCK                : 0
                                |_[06] BIST1LOCK                : 1
                                |_[05] PWRON                    : 0
                                |_[04] PROGFAIL                 : 0
                                |_[03] BUSY                     : 0
                                |_[02] INVALID                  : 0
                                |_[01] FULLDBG                  : 0
                                |_[00] SECURE                   : 1

BSEC_OTP_LOCK                  :0x00000002
                                |_[04] GPLOCK                   : 0
                                |_[03] FENREG                   : 0
                                |_[02] DENREG                   : 0
                                |_[00] OTP                      : 0

BSEC_DENABLE                   :0x0000047f
                                |_[10] DBGSWENABLE              : 1
                                |_[09] CFGSDISABLE              : 0
                                |_[08] CP15SDISABLE[1]          : 0
                                |_[07] CP15SDISABLE[0]          : 0
                                |_[06] SPNIDEN                  : 1
                                |_[05] SPIDEN                   : 1
                                |_[04] HDPEN                    : 1
                                |_[03] DEVICEEN                 : 1
                                |_[02] NIDEN                    : 1
                                |_[01] DBGEN                    : 1
                                |_[00] DFTEN                    : 1

BSEC_FENABLE                   :0x00000000
```

```
                                |_[03] CAN_disable                : 0
                                |_[02] GPU_disable                : 0
                                |_[01] Dual_A7_disable            : 0
                                |_[00] Crypto_disable             : 0


OTP LOWER SHADOW REGISTERS:

ID     value     disturbed     error     R_SLock     W_SLock     Prog_SL     Perm_L

00  0x00000017       0           0           1           1           0           0
01  0x00008000       0           0           0           1           0           1
02  0x00000000       0           0           0           1           1           0
03  0x00000000       0           0           0           0           0           0
04  0x00000000       0           0           0           0           0           0
05  0x00000000       0           0           0           0           0           0
06  0x00000000       0           0           0           0           0           0
07  0x00000000       0           0           0           0           0           0
08  0x00000000       0           0           0           1           0           0
09  0x00000000       0           0           0           0           0           0
10  0x00000000       0           0           0           0           0           0
11  0x00000000       0           0           0           0           0           0
12  0x7ce7f0ee       0           0           0           0           0           1
13  0x00250025       0           0           0           0           0           1
14  0x33385105       0           0           0           0           0           1
15  0x34383330       0           0           0           0           0           1
16  0x125575aa       0           0           0           1           1           1
17  0x1f41185b       0           0           0           0           0           1
18  0x7a200140       0           0           1           1           0           0
19  0x0690147c       0           0           0           0           0           1
20  0x5de00048       0           0           0           0           0           1
21  0x00000000       0           0           0           0           0           1
22  0x00000000       0           0           0           0           0           1
23  0x3fec2fd7       0           0           0           0           0           1
24  0x00000000       0           0           0           1           0           0
25  0x00000000       0           0           0           1           0           0
26  0x00000000       0           0           0           1           0           0
27  0x00000000       0           0           0           1           0           0
28  0x00000000       0           0           0           1           0           0
29  0x00000000       0           0           0           1           0           0
30  0x00000000       0           0           0           1           0           0
31  0x00000000       0           0           0           1           0           0

OTP UPPER SHADOW REGISTERS:

ID     value     disturbed     error     R_SLock     W_SLock     Prog_SL     Perm_L

32  0x00000000       0           0           1           1           1           1
33  0x00000000       0           0           1           1           1           1
34  0x00000000       0           0           1           1           1           1
35  0x00000000       0           0           1           1           1           1
36  0x00000000       0           0           1           1           1           1
37  0x00000000       0           0           1           1           1           1
38  0x00000000       0           0           1           1           1           1
39  0x00000000       0           0           1           1           1           1
40  0x00000000       0           0           0           1           1           1
41  0x00000000       0           0           0           1           1           1
42  0x00000000       0           0           0           1           1           1
43  0x00000000       0           0           0           1           1           1
44  0x00000000       0           0           0           1           1           1
45  0x00000000       0           0           0           1           1           1
46  0x00000000       0           0           0           1           1           1
47  0x00000000       0           0           0           1           1           1
48  0x00000000       0           0           0           1           1           1
49  0x00000000       0           0           0           1           1           1
50  0x00000000       0           0           0           1           1           1
51  0x00000000       0           0           0           1           1           1
```

```
52  0x00000000      0           0           0           1           1           1
53  0x00000000      0           0           0           1           1           1
54  0x00000000      0           0           0           1           1           1
55  0x00000000      0           0           0           1           1           1
56  0x00000000      0           0           1           1           0           0
57  0x42e18000      0           0           0           0           0           0
58  0x0000fd51      0           0           0           0           0           0
59  0x12722301      0           0           0           0           0           0
60  0x00000000      0           0           0           0           0           0
61  0x00000000      0           0           0           0           0           0
62  0x00000000      0           0           0           0           0           0
63  0x00000000      0           0           0           0           0           0
64  0x00000000      0           0           0           0           0           0
65  0x00000000      0           0           0           0           0           0
66  0x00000000      0           0           0           0           0           0
67  0x00000000      0           0           0           0           0           0
68  0x00000000      0           0           0           0           0           0
69  0x00000000      0           0           0           0           0           0
70  0x00000000      0           0           0           0           0           0
71  0x00000000      0           0           0           0           0           0
72  0x00000000      0           0           0           0           0           0
73  0x00000000      0           0           0           0           0           0
74  0x00000000      0           0           0           0           0           0
75  0x00000000      0           0           0           0           0           0
76  0x00000000      0           0           0           0           0           0
77  0x00000000      0           0           0           0           0           0
78  0x00000000      0           0           0           0           0           0
79  0x00000000      0           0           0           0           0           0
80  0x00000000      0           0           0           0           0           0
81  0x00000000      0           0           0           0           0           0
82  0x00000000      0           0           0           0           0           0
83  0x00000000      0           0           0           0           0           0
84  0x00000000      0           0           0           0           0           0
85  0x00000000      0           0           0           0           0           0
86  0x00000000      0           0           0           0           0           0
87  0x00000000      0           0           0           0           0           0
88  0x00000000      0           0           0           0           0           0
89  0x00000000      0           0           0           0           0           0
90  0x00000000      0           0           0           0           0           0
91  0x00000000      0           0           0           0           0           0
92  0x00000000      0           0           0           0           0           0
93  0x00000000      0           0           0           0           0           0
94  0x00000000      0           0           0           0           0           0
95  0x00000000      0           0           0           0           0           0

BSEC_HWCFGR                     :0x00000014
                                |_[07] ECC_USE[3]               : 0
                                |_[06] ECC_USE[2]               : 0
                                |_[05] ECC_USE[1]               : 0
                                |_[04] ECC_USE[0]               : 1
                                |_[03] SAFMEM_SIZE[3]           : 0
                                |_[02] SAFMEM_SIZE[2]           : 1
                                |_[01] SAFMEM_SIZE[1]           : 0
                                |_[00] SAFMEM_SIZE[0]           : 0


BSEC IP REV                     :1.1

BSEC IP ID                      :0x00100032

BSEC IP MAGIC ID                :0xa3c5dd04
```

5.8.3 Update value

It is now possible to use programmer to manage new fuses values: bit to bit for Lower, only programmable once for Upper.

For example: Set the value 0x21458000 into the word 57 (0x39)

```
PC $> STM32_Programmer_CLI -c port=usb1 -otp program wordID=0x39 value=0x21458000
```

Words are written as a 32bit word. It is important to take care of the bit order once using the value in software.

Example: Mac address 00:80:e1:42:45:e5 must be written as:

- Word 57 : 0x42e18000
- Word 58 : 0x0000e545

Warning

MAC address / board Id OTP write should not be performed on ST boards ! They are written and protected by ECC during factory production. Any further write will break the board with no way to recover if these OTP are not locked

5.8.4 Update lock

Lock (no other programmation could occured) could be added per word; the permanent lock access will only be refreshed after boot.

For example: permanent lock (pl=1) the word 57 (0x39)

```
PC $> STM32_Programmer_CLI -c port=usb1 -otp program wordID=0x39 value=0x21458000 pl=1
```

5.9 How to program STPMIC NVM

STM32CubeProgrammer require to be connected to U-Boot in order to access STPMIC NVM alternate (0xF4).

Refer to previous Chapter 5.8.1 and follow same procedure to flash only the 2 first "boot" partition with GUI. Then disconnect from GUI and switch to CLI in a terminal.

This is procedure to modify NVM values.

- Read partition of pmic and load it into a .bin file

```
PC $> STM32_Programmer_CLI -c port=usb1  -rp 0xf4 0x0 0x8 $MySelectedPATH\PMIC_NVM_read.
bin
```

- Backup it by creating a copy PMIC_NVM_write.bin
- Use a binary editor in HEX format (eg "vi -b file.bin" then :%!xxd) to edit the copy PMIC_NVM_write.bin with new values.

The binary format display the shadow register from 0xF8 to 0XFF. Refer to STPMIC1 DataSheet DS12792.

```
00000000: EE92 C002 F280  0233                     .......3
respectively SHR address :  F8F9 FAFB FCFD FEFF
```

- Program back the STPMIC1 with modified NVM using following command:

```
PC $> STM32_Programmer_CLI -c port=usb1 -pmic "PMIC_NVM_write.bin"
```

Universal Asynchronous Receiver/Transmitter

former spelling for e•MMC ('e' in italic)

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Linux® is a registered trademark of Linus Torvalds.

Device Firmware Upgrade

Operating System

High Speed (MIPI® Alliance DSI standard)

Microprocessor Unit

System Trace Module

First Stage Boot Loader

Second Stage Boot Loader

One Time Programmed

Das U-Boot -- the Universal Boot Loader (see U-Boot_overview)

Trusted Firmware for Arm Cortex-A

Graphical User Interface

Central processing unit

Cortex®

Boot and Security and OTP control

Controller Area Network (robust bus mainly used for automotive applications)

Graphics Processing Units

Elliptic curve cryptography

Error Correction Capability

media access control address (https://en.wikipedia.org/wiki/MAC_address)

Non Volatile Memory, like a flash memory

Power Management Integrated Circuit

Stable: 23.06.2020 - 14:55 / Revision: 12.06.2020 - 10:12

A quality version of this page, approved on *23 June 2020*, was based off this revision.

Click on the link below that corresponds to your targeted distribution, to discover the software Packages (Starter, Developer and Distribution) delivered for the STM32 MPU microprocessor devices:

Which STM32MPU Embedded Software Package better suits your needs

Which STM32MPU Embedded Software Package for Android better suits your needs

Microprocessor Unit