PC prerequisites

Stable: 11.10.2019 - 12:30 / Revision: 11.10.2019 - 08:10

Recommended setup: Native Linux PC

Contents

1 Purpose 1
2 Recommended PC configurations 1
3 Linux PC 2
3.1 Check Internet access 2
3.2 Install extra packages 4
3.2.1 Install extra packages for Android 5
3.3 Additional configurations 5
3.4 Setup *Git* user information 6
4 Windows PC 7
4.1 Virtual Machine System 7
4.1.1 Virtual Machine installation 7
4.1.2 Download the Ubuntu image for the virtual machine 7
4.1.3 Launch of Ubuntu image 8
4.2 WSL2 (experimental) 9
4.3 References 10

1 Purpose

This article explains and describes the hardware configuration required to be able 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 Windows7 and Windows10 Preferred version Windows 10	native	Virtual Machine	Virtual Machine
Linux (64 bits) Tested with Ubuntu 18.04 and 16.04	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 configurations when using *Developer Package* or *Distribution Package*.

The table below correspond to the minimal validated configuration:

Hardware item	**Minimal validated configuration**	**Comments / Recommendations**
CPU	core i5-2540M @ 2.6GHz 2 cores (4 threads) 3MB cache	**64 bits instruction set is mandatory** **8 cores/threads or more** is a good config moreover for *Developer Package* and *Distribution Package*.
RAM	8GB	**16GB or more** is recommended especially for *Virtual Machine* setup , *Developer Package* and *Distribution Package*.
Hard Drive	320GB	**1TB** is probably a better config when using *Distribution Package*

3 Linux PC

A Linux PC with **Ubuntu 16.04** or **Ubuntu 18.04** is the recommended setup. Other Ubuntu revisions should also be supported, please refer to Yocto Manual[1] .

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

3.1 Check Internet access

- **An Internet access through http and https protocols must be provided.**

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

The command below allows to check for Internet access through http/https protocols:

```
PC $> wget -q www.google.com && echo "Internet access over HTTP/HTTPS is OK !" || echo "No
```

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

Any other situation likely indicates the need for a proxy 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
PC $> export https_proxy=http://<MyProxyLogin>:<MyProxyPassword>@<MyProxyServerUrl>:<MyPro
```

Because your password may contains "special characters" you need to translate your password into ASCII hexacode. By this way you can translate <MyProxyPassword> into hexacode by using this command :

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

Check again the Internet access with command:

```
PC $> wget -q www.google.com && echo "Internet access over HTTP/HTTPS is OK !" || echo "No
```

Internet access for *sudo* commands

Required for *Distribution Package*.

sudo commands are executed in the *root* user environment; by default, no Internet proxy settings are applied for *root* user.
Root user should be able to browse Internet, after creation of an alias passing the proxy settings on *sudo* command:

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

Check that the *sudo* commands is 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), some other protocols like git:// or ssh:// may be required.

For example in the context of the *Distribution Package*, some "git fetch" commands could require "git:// protocols".
In order to support these protocols through a proxy, the best way is to directly setup the proxy in the $HOME/.gitconfig file (core.gitproxy) and use a tool like *cockscrew*[2] in order 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>" (optionn
PC $> echo 'exec corkscrew <MyProxyServerUrl> <MyProxyPort> $* $HOME/.git-proxy.auth' > $H
PC $> chmod 700 $HOME/bin/git-proxy.sh
PC $> echo '<MyProxyLogin>:<MyProxyPassword>' > $HOME/.git-proxy.auth
PC $> chmod 600 $HOME/.git-proxy.auth
```

Here is a command to test this proxy settings:

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

The command should return 'OK', else proxy settings are wrong.

3.2 Install extra packages

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

In order to do basic development tasks, basic cross-compilation (via *Developer Package*) or more complex cross-compilation as OpenEmbedded does (via *Distribution Package*), some extra Ubuntu packages should be installed:

```
PC $> sudo apt-get update
PC $> sudo apt-get install bison flex sed wget curl cvs subversion git-core coreutils unzi
```

```
PC $> sudo apt-get install libsdl1.2-dev xterm corkscrew nfs-common nfs-kernel-server devi
PC $> sudo apt-get install ncurses-dev bc linux-headers-generic gcc-multilib libncurses5-d
```

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

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

3.2.1 Install 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-uti
```

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

- Allow up to 16 partitions per mmc

By default, on Linux system, a maximum of 8 partitions are allowed on mmc. All Packages (Starter Package, ...) need more than 10 partitons for the storage device. In order 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
```

- Check 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
```

In case the *locale* command returns a different configuration than the one shown above, it must be reconfigured as follows:

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

- Add user in basics groups

The *user* login should belong to the basic Linux groups such as **disk**, **tty**, **dialout** or **plugdev**
Use the command *groups* 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 Setup *Git* user information

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

The User Information is needed by git[3] in case *commit* and/or *push* commands are being 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.**

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 bits** Ubuntu image compatible with your virtual machine

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

4.1 Virtual Machine System

4.1.1 Virtual Machine installation

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 Download the Ubuntu image for the virtual machine

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

Setup have been validated and tested on Ubuntu 16.04 (64bit) and Ubuntu 18.04 (64bit).

Download the 64 bits Ubuntu image available at [8] and:

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 be
- Network: NAT is good and an easy way to benefit form a network connection within the vir
```

Virutal size of virtual disk downloaded from osboxes.org is about 500GB. Even if the real size of the file of the virtual disk is less at beginning, the size could growth up to 500GB over compiling distribution package or development package.

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

Please refer to the VMwarePlayer screenshot tutorial.

4.1.3 Launch of Ubuntu image

For "AZERTY" keyboard users:

The default keyboard configuration is "QWERTY".
In order to configure the keyboard for "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 opened, 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 **Credentials** of the Ubuntu **are set to "osboxes.org" for both login and password.**

Adjust screen resolution:

The (default) resolution used by the virtual machine is 800x600 (smallest available). It is not automatically adjusted to the display resolution. In order 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 the document: PreRequisite-Vmware-tools.pdf

The virtual machine is up and running!

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

USB connection's speed:

USB connection is requested for accessing 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 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, we suggest to:

- **try different USB configurations of the virtual machine;**
- **connect the USB device directly on 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* works in WSL2 (Windows Sub-system Linux 2) environment. WSL is a feature provided by **Windows 10®**.
ST has run unsuccessfully *Developer Package* and *Distribution Package* on WSL but successfully on WSL2.
WSL2 is available on Windows 10® since build 18917.

WSL 2 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).

- WSL2 - Installation :
  - To install WSL2 please read this webpage: https://docs.microsoft.com/fr-fr/windows/wsl/wsl2-install
  - Once WSL2 installed, jump to chapter #Linux_PC to make your WSL2 ready to run *Developer Package* and/or *Distribution Package*.
- WSL2 - Limitations :
  - WSL2 up to now (09/2019) does not support hardware such as USB devices, serial, ... (more details).

    This means, STM32CubeProgrammer should be used through native Windows
  - WSL2 files are not browsable from Windows native file explorer.

    To share files between WSL2 and Windows, the prefered way is to use the mount point */mnt/c* from WSL2 and do copies.

- WSL2 - Tips :
  - Launch graphical application : On *wiki.ubuntu.com* the page on WSL contains a chapter Running Graphical Applications.

4.3 References

1. ↑ https://www.yoctoproject.org/docs/2.4.3/ref-manual/ref-manual.html#detailed-supported-distros
2. ↑ https://en.wikipedia.org/wiki/Corkscrew_(program)
3. ↑ Git
4. ↑ http://vmware.com
5. ↑ https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/15_0
6. ↑ https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html
7. ↑ http://osboxes.org
8. ↑ http://www.osboxes.org/ubuntu/#ubuntu-16-04-vmware

Random Access Memory