



How to configure uniprocessor mode



Contents

1. How to configure uniprocessor mode	3
2. Menuconfig or how to configure kernel	6
3. STM32MP15 Linux kernel overview	12



STMicroelectronics Confidential

A quality version of this page, approved on *3 February 2020*, was based off this revision.



1 Purpose

This article explains how to switch from a SMP (symmetric multiprocessing) Linux[®] kernel configuration to a UP (uniprocessor) Linux kernel configuration.



2 Rationale

Considering that STMicroelectronics Linux kernel is configured in SMP mode by default, single Arm Cortex-A core users may wish to disable this configuration in order to avoid trying to start the second core at every boot. Please note that keeping the SMP mode enabled on a single processor does not have any performance impact on the kernel in most use cases, so this change is not mandatory.



3 Modify kernel configuration

To switch to UP configuration, disable SMP configuration. To do that, search **SMP** in kernel menuconfig and disable the option.

symetric multiprocessing

Linux[®] is a registered trademark of Linus Torvalds.

uniprocessor

Arm[®] is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere. 

Cortex[®]
Stable: 11.02.2021 - 11:10 / Revision: 19.01.2021 - 10:34

A quality version of this page, approved on 11 February 2021, was based off this revision.

Contents

1 Linux configuration genericity	7
2 Menuconfig and Developer Package	9
3 Menuconfig and Distribution Package	11
4 References	12



1 Linux configuration genericity

The process of building a kernel has two parts: configuring the kernel options and building the source with those options.

The Linux[®] kernel configuration is found in the generated file: `.config`.

`.config` is the result of configuring task which is processing platform `defconfig` and fragment files if any.

For OpenSTLinux distribution the `defconfig` is located into the kernel source code and fragments into `stm32mp` BSP layer :

- `arch/arm/configs/multi_v7_defconfig`

Every new kernel version brings a bunch of new options, we do not want to back port them into a specific `defconfig` file each time the kernel releases, so we use the same `defconfig` file based on ARM SoC v7 architecture.

STM32MP1 specificities are managed with fragments `config` files.

- `meta-st/meta-st-stm32mp/recipes-kernel/linux/linux-stm32mp/<kernel version>/fragment-*.config`

`.config` result is located in the build folder:

- `build-openstlinuxweston-stm32mp1/tmp-glibc/work/stm32mp1-ostl-linux-gnueabi/linux-stm32mp/4.14-48/linux-stm32mp1-standard-build/.config`

To modify the kernel options, it is not recommended to edit this file directly.

- A user runs either a text-mode :

```
PC $> make config
starts a character based question and answer session (Figure 1)
```

```
[greg@shamp linux-2.5]$ make config
make[1]: `scripts/kconfig/conf' is up to date.
./scripts/kconfig/conf arch/i386/Kconfig
#
# using defaults found in .config
#
*
* Linux Kernel Configuration
*
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers (EXPERIMENTAL) [Y/n/?] █
```

Figure 1. Configuring the kernel with make config

```
PC $> make
menuconfig
starts a terminal-
oriented
configuration tool
(using ncurses)
(Figure 2)
The ncurses text
version is more
popular and is run
with the make
menuconfig option.
Wikipedia Menuconfig[1]
] also explains how
to "navigate" within
the configuration
menu, and highlights
main key strokes.
```

configurator :

- or a graphical kernel

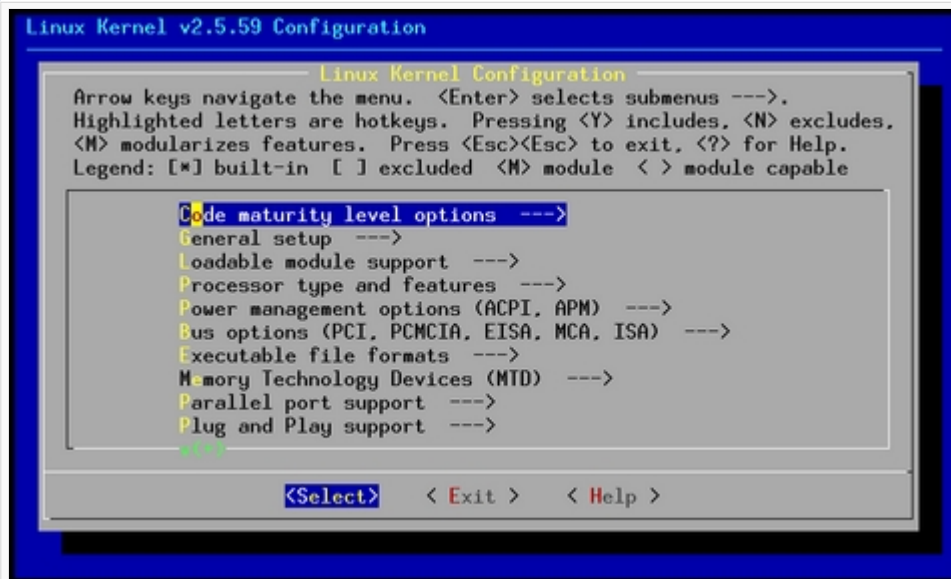


Figure 2. Make menuconfig makes it easier to back up and correct mistakes

PC \$> make xconfig starts a X based configuration tool (Figure 3)

Ultimately these configuration tools edit the .config file.

An option indicates either some driver is built into the kernel ("=y") or will be built as a module ("=m") or is not selected.

The unselected state can either be indicated by a line starting with "#" (e.g. "# CONFIG_SCSI is not set") or by the absence of the relevant line from the .config file.

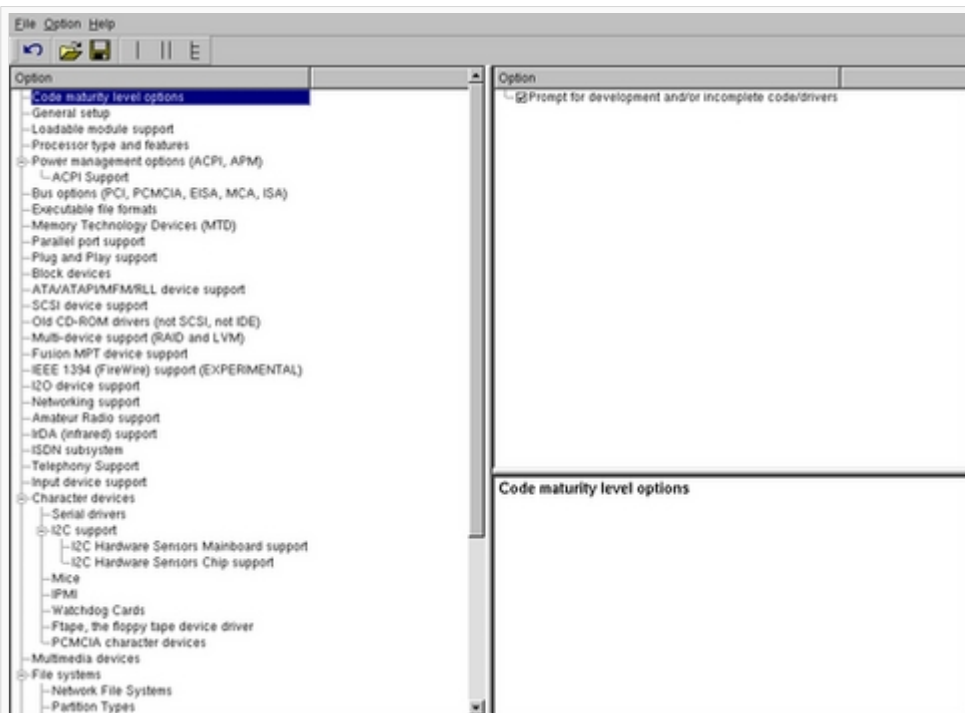


Figure 3. The Qt-Based make xconfig

The 3 states of the main selection option for the SCSI subsystem (which actually selects the SCSI mid level driver) follow. Only one of these should appear in an actual .config file:

```
CONFIG_SCSI=y
CONFIG_SCSI=m
# CONFIG_SCSI is not set
```




2 Menuconfig and Developer Package

For this use case, the prerequisite is that OpenSTLinux SDK has been installed and configured.

To verify if your cross-compilation environment has been put in place correctly, run the following command:

```
PC $> set | grep CROSS
CROSS_COMPILE=arm-ostl-linux-gnueabi-
```

For more details, refer to <Linux kernel installation directory>/README.HOW_TO.txt helper file (the latest version of this helper file is also available in GitHub: [README.HOW_TO.txt](#)).

- Go to the <Linux kernel build directory>

```
PC $> cd <Linux kernel build directory>
```

- Save initial configuration (to identify later configuration updates)

```
PC $> make arch=ARM savedefconfig
Result is stored in defconfig file
PC $> cp defconfig defconfig.old
```

- Start the Linux kernel configuration menu

```
PC $> make arch=ARM menuconfig
```

- Navigate forwards or backwards directly between feature
 - un/select, modify feature(s) you want
 - When the configuration is OK : exit and save the new configuration

```
useful keys to know:
enter: enter in config subdirectory
space: hit several times to either select [*], select in module [m] or unselect [ ]
/: to search for a keyword, this is usefull to navigate in tree
?: to have more information on selected line
```

- Compare the old and new config files after operating modifications with menuconfig

```
PC $> make arch=ARM savedefconfig
```

Retrieve configuration updates by comparing the new defconfig and the old one

```
PC $> meld defconfig defconfig.old
```

- Cross-compile the Linux kernel (please check the load address in the *README.HOW_TO.txt* helper file)



```
PC $> make arch=ARM uImage LOADADDR=<loadaddr of kernel>  
PC $> cp arch/arm/boot/uImage install_artifact/boot/
```

- Update the Linux kernel image on board

```
PC $> scp install_artifact/boot/uImage root@<board ip address>:/boot/
```

Information

If the `/boot` mounting point doesn't exist yet, please see [how to create a mounting point](#)

- Reboot the board

```
Board $> cd /boot; sync; systemctl reboot
```

Note that this use case modifies the configuration file in the Linux kernel build directory, not in the Linux kernel source directory: this is a temporary modification useful for a prototyping.

- To make this temporary modification permanent, the delta between `defconfig` and `defconfig.old` must be saved in a configuration fragment file (`fragment-*.config`) based on `fragment.cfg` file, and the Linux kernel configuration/compilation steps must be re-executed (as explained in the `README.HOW_TO.txt` helper file).



3 Menuconfig and Distribution Package

- Start the Linux kernel configuration menu

```
PC $> bitbake virtual/kernel -c menuconfig
```

- Navigate forwards or backwards directly between feature
 - un/select, modify feature(s) you want
 - When the configuration is OK : exit and save the new configuration

```
useful keys to know:
enter: enter in config subdirectory
space: hit several times to either select [*], select in module [m] or unselect [ ]
/: to search for a keyword, this is usefull to navigate in tree
?: to have more information on selected line
```

- Cross-compile the Linux kernel

```
PC $> bitbake virtual/kernel
```

- Update the Linux kernel image on board

```
PC $> scp <build dir>/tmp-glibc/deploy/images/<machine name>/uImage root@<board ip address>:/boot
```

Information

If the `/boot` mounting point does not exist yet, please see [how to create a mounting point](#)

- Reboot the board

```
Board $> cd /boot; sync; systemctl reboot
```

Note that this use case modifies the configuration file in the Linux kernel build directory, not in the Linux kernel source directory: this is a temporary modification useful for a prototyping.

- To make this temporary modification permanent, it must be saved in a configuration fragment file (fragment-*.config) based on `fragment.cfg` file, and the Linux kernel configuration/compilation steps must be re-executed: `bitbake <name of kernel recipe>`.



4 References

- [Wikipedia Menuconfig](#)

Linux[®] is a registered trademark of Linus Torvalds.

Board support package

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

A quality version of this page, approved on 11 January 2021, was based off this revision.

This section gives an overview of **Linux[®] kernel drivers** (UPPERCASE in the figure) implemented for the STM32MP15 support, with their respective **software frameworks** (lowercase in the figure).

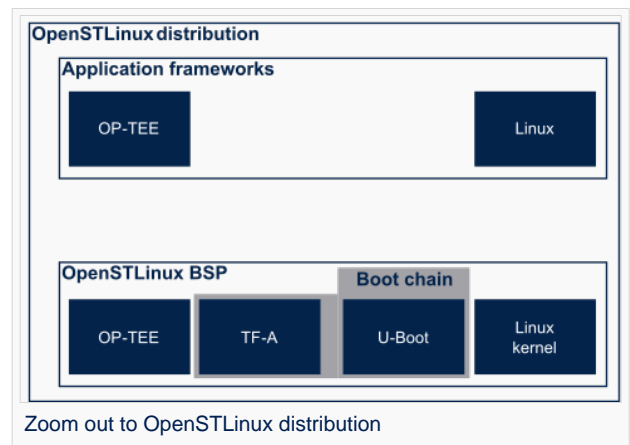
The components are grouped per **functional domains**.

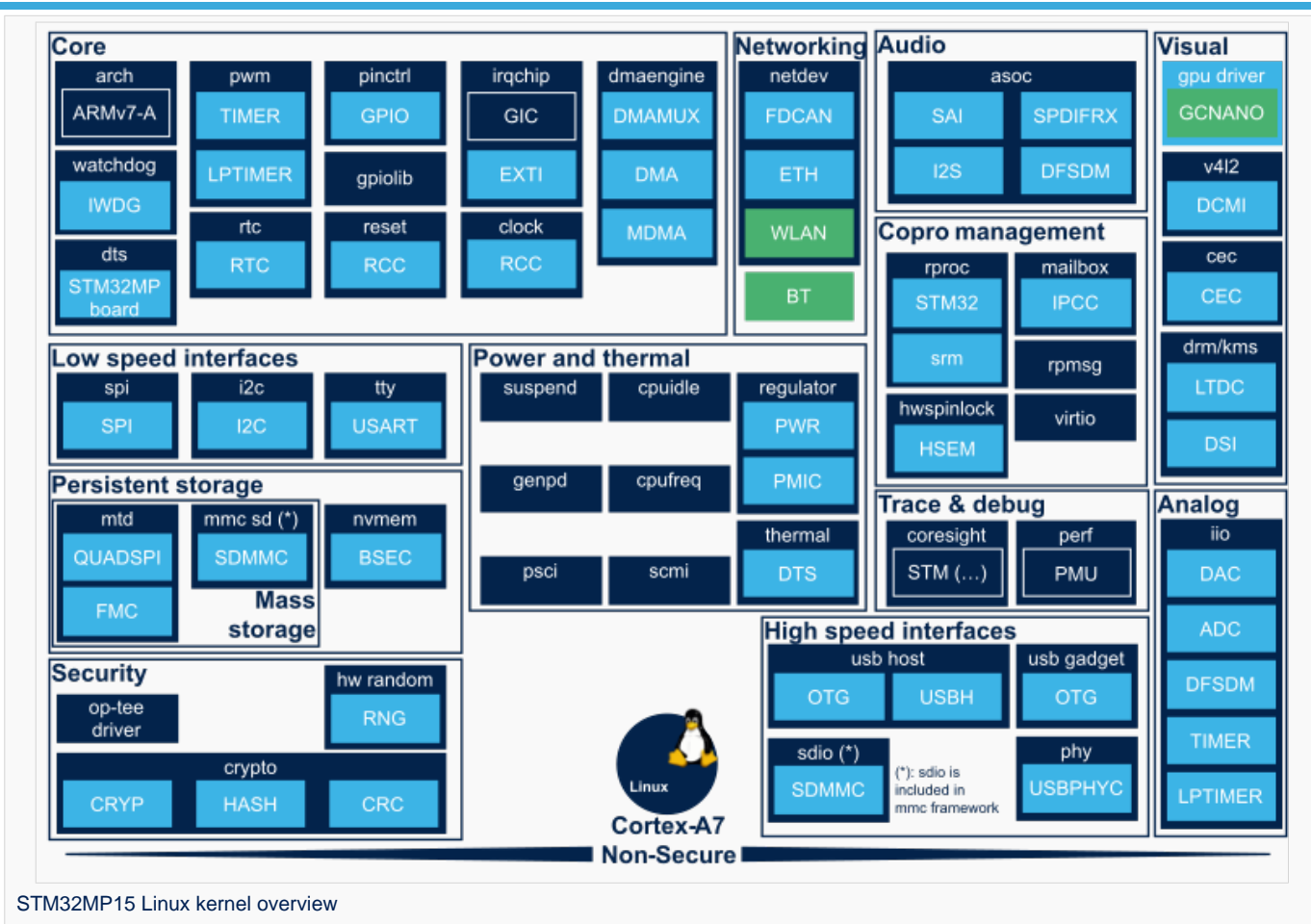
Each **Linux framework** is further described in [Linux operating system category articles](#).

Each STM32 MPU **peripheral** is introduced in [peripherals overview articles](#).

Both those sections are reusing the same functional domain split.

The color code, explained in the legend, allows to see the code origin for each component.





STM32MP15 Linux kernel overview

Linux® is a registered trademark of Linus Torvalds.

Microprocessor Unit