



Debugfs



Contents

1. Debugfs	3
2. Menuconfig or how to configure kernel	18
3. Pseudo filesystem	25
4. STM32MP15 Linux kernel overview	34



A quality version of this page, approved on 4 February 2020, was based off this revision.

Contents

1 Introduction	4
2 Installing debugfs on your target board	5
3 Getting started	6
3.1 How to mount debugfs	6
3.2 How to use debugfs in Linux kernel drivers	6
3.3 How to visualize debugfs information	6
4 To go further	15
4.1 Regmap (register map) cache	15
5 References	18



1 Introduction

Debugfs is a simple-to-use RAM-based file system, specially designed for debugging purposes.

It is provided as a simple way for kernel developers to make information available to the user space.

It is part of a *pseudo filesystem*. Unlike *procfs*, which is only meant for process information, or *sysfs*, which has strict one-value-per-file rules, *debugfs* has no rules at all.

Developers are free to put any information into *debugfs*.



2 Installing debugfs on your target board

Debugfs is enabled and ready to be used in all STM32MPU Embedded Software distributions, via the Linux[®] kernel configuration **CONFIG_DEBUG_FS**, set to yes by default.

```
Symbol: DEBUG_FS  
Location:  
  Kernel Hacking --->  
    Compile-time checks and compiler options -->  
      [*] Debug Filesystem
```

Please refer to the [Menuconfig or how to configure kernel](#) article for instructions on modifying the configuration and recompiling the Linux kernel image in the distribution package context.



3 Getting started

3.1 How to mount debugfs

Debugfs is typically mounted at `/sys/kernel/debug` with a command such as:

```
Board $> mount -t debugfs none /sys/kernel/debug
```

3.2 How to use debugfs in Linux kernel drivers

Debugfs is a simple way for kernel drivers to make information available to user space.

It can be manipulated using several calls from the C header file `include/linux/debugfs.h`, such as:

```
debugfs_create_file → for creating a file in the debug filesystem.
debugfs_create_dir → for creating a directory in the debug filesystem.
debugfs_create_symlink → for creating a symbolic link in the debug filesystem.
debugfs_create_u32 → for creating a file containing a single 32-bit unsigned integer value
debugfs_remove → for removing a debugfs entry from the debug filesystem.
...
```

For further details, refer to the Linux kernel documentation for debugfs^[1].

For example, you can refer to an OpenSourceForU article: ^[2].

3.3 How to visualize debugfs information

On the target board, go the debugfs mount directory:

```
Board $> cd /sys/kernel/debug
```

There you will find all the debugfs entries created by Linux kernel frameworks

```
Board $> ls -l
```

```
drwxr-xr-x  2 root root 0 Jan  1  1970 49000000.usb-otg
drwxr-xr-x  4 root root 0 Jan  1  1970 asoc
drwxr-xr-x 40 root root 0 Jan  1  1970 bdi
drwxr-xr-x 10 root root 0 Jan  1  1970 block
drwxr-xr-x  2 root root 0 Jan  1  1970 bluetooth
drwxr-xr-x  3 root root 0 May 28 15:52 cec
drwxr-xr-x 225 root root 0 Jan  1  1970 clk
drwxr-xr-x  2 root root 0 Jan  1  1970 dma_buf
drwxr-xr-x  3 root root 0 Jan  1  1970 dri
drwxr-xr-x  2 root root 0 Jan  1  1970 dynamic_debug
drwxr-xr-x  2 root root 0 Jan  1  1970 extfrag
-rw-r--r--  1 root root 0 Jan  1  1970 fault_around_bytes
drwxr-xr-x  3 root root 0 May 28 15:52 gc
-r--r--r--  1 root root 0 Jan  1  1970 gpio
```



```

drwxr-xr-x 2 root root 0 Jan 1 1970 hid
drwxr-xr-x 2 root root 0 Jan 1 1970 ieee80211
drwxr-xr-x 7 root root 0 Jan 1 1970 iio
-r--r--r-- 1 root root 0 Jan 1 1970 irq_domain_mapping
drwxr-xr-x 2 root root 0 Jan 1 1970 memblock
drwxr-xr-x 3 root root 0 Jan 1 1970 mmc0
drwxr-xr-x 3 root root 0 Jan 1 1970 mmc1
drwxr-xr-x 11 root root 0 Jan 1 1970 mtd
drwxr-xr-x 4 root root 0 Jan 1 1970 pinctrl
drwxr-xr-x 4 root root 0 Jan 1 1970 pm_genpd
drwxr-xr-x 2 root root 0 Jan 1 1970 pm_qos
-r--r--r-- 1 root root 0 Jan 1 1970 pwm
drwxr-xr-x 2 root root 0 Jan 1 1970 ras
drwxr-xr-x 13 root root 0 Jan 1 1970 regmap
drwxr-xr-x 21 root root 0 Jan 1 1970 regulator
drwxr-xr-x 3 root root 0 Jan 1 1970 remoteproc
-r--r--r-- 1 root root 0 Jan 1 1970 sleep_time
drwxr-xr-x 3 root root 0 Jan 1 1970 stmmaceth
-r--r--r-- 1 root root 0 Jan 1 1970 suspend_stats
dr-xr-xr-x 3 root root 0 Jan 1 1970 tracing
drwxr-xr-x 2 root root 0 Jan 1 1970 ubi
drwxr-xr-x 2 root root 0 Jan 1 1970 ubifs
drwxr-xr-x 4 root root 0 Jan 1 1970 usb
-r--r--r-- 1 root root 0 Jan 1 1970 wakeup_sources

```

You can then browse to find the requested information.

Please note that the different Linux software frameworks available in [STM32MP15 Linux kernel overview](#) provide specific trace and debug information in a "How to trace and debug the framework" dedicated chapter, including debugfs.

Some examples:

- To list the wakeup sources:

```

Board $> cat /sys/kernel/debug/wakeup_sources
name          active_count  event_count  wakeup_count  expire_count
active_since  total_time   max_time     last_change   prevent_suspend_time
5c002000.i2c:stpmul@33:onkey  0             0             0             4075           0
0             0             0             0             0
0-0033        0             0             0             0
0             0             0             3949          0
alarmtimer    0             0             0             0
0             0             0             3901          0
5c004000.rtc  1             1             0             0
0             0             0             5246          0
deleted       0             0             0             0
0             0             0             0             0

```

- To obtain a summary of the clock settings:

```

Board $> cat /sys/kernel/debug/clk/clk_summary
clock          enable_cnt  prepare_cnt  rate  accuracy  phase
-----
ck_usbo_48m    1           1           48000000  0 0
  usbo_k       1           1           48000000  0 0
ck_dsi_phy     1           1             0      0 0
  dsi_k        1           1             0      0 0
i2s_ckin       0           0             0      0 0
clk-csi        1           1           4000000  0 0
  ck_csi       1           1           4000000  0 0
  rng2_k       0           0           4000000  0 0

```



rng1_k	0	0	4000000	0 0
clk-lsi	1	1	32000	0 0
ck_lsi	1	1	32000	0 0
dac12_k	0	0	32000	0 0
clk-lse	1	1	32768	0 0
ck_lse	1	1	32768	0 0
ck_rtc	2	2	32768	0 0
rtc_lsco	1	1	32768	0 0
lptim5_k	0	0	32768	0 0
lptim4_k	0	0	32768	0 0
cec_k	0	0	32768	0 0
clk-hsi	1	1	64000000	0 0
clk-hsi-div	1	1	64000000	0 0
ck_hsi	2	2	64000000	0 0
ck_mco1	0	0	64000000	0 0
uart8_k	0	0	64000000	0 0
uart7_k	0	0	64000000	0 0
uart6_k	0	0	64000000	0 0
uart5_k	0	0	64000000	0 0
uart4_k	1	1	64000000	0 0
usart3_k	0	0	64000000	0 0
usart2_k	1	1	64000000	0 0
usart1_k	0	0	64000000	0 0
i2c6_k	0	0	64000000	0 0
i2c4_k	0	0	64000000	0 0
i2c5_k	0	0	64000000	0 0
i2c3_k	0	0	64000000	0 0
i2c2_k	0	0	64000000	0 0
i2c1_k	0	0	64000000	0 0
spi6_k	0	0	64000000	0 0
spi5_k	0	0	64000000	0 0
spi4_k	0	0	64000000	0 0
clk-hse	1	1	24000000	0 0
ck_hse	7	7	24000000	0 0
ck_hse_rtc	0	0	1000000	0 0
stgen_k	1	1	24000000	0 0
usbphy_k	1	1	24000000	0 0
ck_per	1	1	24000000	0 0
adc12_k	1	1	24000000	0 0
ref4	1	1	24000000	0 0
pll4	1	1	594000000	0 0
pll4_r	0	0	74250000	0 0
pll4_q	1	1	31263158	0 0
ltdc_px	1	1	31263158	0 0
dsi_px	0	0	31263158	0 0
fdcan_k	0	0	31263158	0 0
pll4_p	0	0	74250000	0 0
ref3	1	1	24000000	0 0
pll3	2	3	786431640	0 0
pll3_r	1	1	98303955	0 0
sdmmc3_k	0	0	98303955	0 0
sdmmc2_k	1	1	98303955	0 0
sdmmc1_k	0	0	98303955	0 0
pll3_q	0	1	49151978	0 0
adfsdm_k	0	0	49151978	0 0
sai4_k	0	0	49151978	0 0
sai3_k	0	0	49151978	0 0
sai2_k	0	2	49151978	0 0
sai1_k	0	0	49151978	0 0
spi3_k	0	0	49151978	0 0
spi2_k	0	0	49151978	0 0
spi1_k	0	0	49151978	0 0
spdif_k	0	0	49151978	0 0
pll3_p	1	1	196607910	0 0
ck_mcu	6	16	196607910	0 0
dfsdm_k	0	0	196607910	0 0
gpiok	0	0	196607910	0 0



gpioj	0	0	196607910	0 0
gpioi	0	1	196607910	0 0
pioh	0	1	196607910	0 0
piog	0	1	196607910	0 0
piof	0	1	196607910	0 0
pioe	0	1	196607910	0 0
piod	0	1	196607910	0 0
pioc	0	1	196607910	0 0
piob	0	1	196607910	0 0
pioa	0	1	196607910	0 0
ipcc	2	2	196607910	0 0
hsem	0	0	196607910	0 0
crc2	0	0	196607910	0 0
rng2	0	0	196607910	0 0
hash2	0	0	196607910	0 0
cryp2	0	0	196607910	0 0
dcmi	0	0	196607910	0 0
sdmmc3	0	0	196607910	0 0
usbo	0	0	196607910	0 0
adc12	1	1	196607910	0 0
dmamux	1	1	196607910	0 0
dma2	0	0	196607910	0 0
dma1	1	1	196607910	0 0
pclk3	1	1	98303955	0 0
lptim3_k	0	0	98303955	0 0
lptim2_k	0	0	98303955	0 0
hdp	0	0	98303955	0 0
pmbctrl	0	0	98303955	0 0
tmpsens	0	0	98303955	0 0
vref	0	0	98303955	0 0
syscfg	1	1	98303955	0 0
sai4	0	0	98303955	0 0
lptim5	0	0	98303955	0 0
lptim4	0	0	98303955	0 0
lptim3	0	0	98303955	0 0
lptim2	0	0	98303955	0 0
pclk2	0	0	98303955	0 0
fdcan	0	0	98303955	0 0
dfsdm	0	0	98303955	0 0
sai3	0	0	98303955	0 0
sai2	0	0	98303955	0 0
sai1	0	0	98303955	0 0
usart6	0	0	98303955	0 0
spi5	0	0	98303955	0 0
spi4	0	0	98303955	0 0
spi1	0	0	98303955	0 0
tim17	0	0	98303955	0 0
tim16	0	0	98303955	0 0
tim15	0	0	98303955	0 0
tim8	0	0	98303955	0 0
tim1	0	0	98303955	0 0
ck2_tim	0	0	196607910	0 0
tim17_k	0	0	196607910	0 0
tim16_k	0	0	196607910	0 0
tim15_k	0	0	196607910	0 0
tim8_k	0	0	196607910	0 0
tim1_k	0	0	196607910	0 0
pclk1	0	2	98303955	0 0
lptim1_k	0	0	98303955	0 0
mdio	0	0	98303955	0 0
dac12	0	0	98303955	0 0
cec	0	0	98303955	0 0
spdif	0	0	98303955	0 0
i2c5	0	0	98303955	0 0
i2c3	0	0	98303955	0 0
i2c2	0	0	98303955	0 0
i2c1	0	0	98303955	0 0



```

uart8          0          0  98303955      0 0
uart7          0          0  98303955      0 0
uart5          0          0  98303955      0 0
uart4          0          0  98303955      0 0
usart3         0          0  98303955      0 0
usart2         0          0  98303955      0 0
spi3           0          0  98303955      0 0
spi2           0          1  98303955      0 0
lptim1        0          0  98303955      0 0
tim14         0          0  98303955      0 0
tim13         0          0  98303955      0 0
tim12         0          0  98303955      0 0
tim7          0          0  98303955      0 0
tim6          0          0  98303955      0 0
tim5          0          0  98303955      0 0
tim4          0          0  98303955      0 0
tim3          0          0  98303955      0 0
tim2          0          0  98303955      0 0
ck1_tim       0          1  196607910     0 0
  tim14_k     0          0  196607910     0 0
  tim13_k     0          0  196607910     0 0
  tim12_k     0          0  196607910     0 0
  tim7_k      0          0  196607910     0 0
  tim6_k      0          1  196607910     0 0
  tim5_k      0          0  196607910     0 0
  tim4_k      0          0  196607910     0 0
  tim3_k      0          0  196607910     0 0
  tim2_k      0          0  196607910     0 0
ref1          2          2   24000000     0 0
pll2         2          2  533000000     0 0
  pll2_r      1          1  533000000     0 0
  pll2_q      0          0  533000000     0 0
  gpu_k       0          0  533000000     0 0
  pll2_p      1          1  266500000     0 0
  ck_axi      8          9  266500000     0 0
  ck_trace   0          0  133250000     0 0
  ck_sys_dbg 0          0  266500000     0 0
  qspi_k     0          0  266500000     0 0
  fmc_k      0          0  266500000     0 0
  ethstp     0          0  266500000     0 0
  usbh       1          1  266500000     0 0
  crcl       0          0  266500000     0 0
  sdmmc2     0          0  266500000     0 0
  sdmmc1     0          0  266500000     0 0
  qspi       0          0  266500000     0 0
  fmc        0          0  266500000     0 0
  ethmac     1          1  266500000     0 0
  ethrx      1          1  266500000     0 0
  ethtx      1          1  266500000     0 0
  gpu        0          0  266500000     0 0
  mdma       1          1  266500000     0 0
  bkpsram    0          0  266500000     0 0
  rng1       0          0  266500000     0 0
  hash1      0          0  266500000     0 0
  cryp1      0          0  266500000     0 0
  gpioz      0          1  266500000     0 0
  tzc2       0          0  266500000     0 0
  tzc1       0          0  266500000     0 0
  pclk5      1          1   66625000     0 0
  stgen      0          0   66625000     0 0
  bsec       0          0   66625000     0 0
  iwdg1      0          0   66625000     0 0
  tzpc       0          0   66625000     0 0
  rtcapb     2          2   66625000     0 0
  usart1     0          0   66625000     0 0
  i2c6       0          0   66625000     0 0
  i2c4       0          0   66625000     0 0

```



spi6	0	0	66625000	0	0
pclk4	1	1	133250000	0	0
stgenro	0	0	133250000	0	0
usbphy	0	0	133250000	0	0
iwdg2	1	1	133250000	0	0
dsi	0	0	133250000	0	0
ltdc	0	0	133250000	0	0
pll1	1	1	650000000	0	0
pll1_p	1	1	650000000	0	0
ck_mpu	1	1	650000000	0	0
ck_mco2	0	0	650000000	0	0
clk-hse-div2	0	0	120000000	0	0
ethptp_k	0	0	0	0	0
ethck_k	0	0	0	0	0

- To obtain a summary of the regulator settings:

```
Board $> cat /sys/kernel/debug/regulator/regulator_summary
regulator          use open bypass voltage current      min      max
-----
regulator-dummy    0  11    0    0mV      0mA      0mV      0mV
49000000.usb-otg   0  0     0    0mV      0mA      0mV      0mV
49000000.usb-otg   0  0     0    0mV      0mA      0mV      0mV
5a000000.dsi.0     0  0     0    0mV      0mA      0mV      0mV
vddcore            0  0     0  1200mV   0mA      800mV   1350mV
vdd_ddr            0  1     0  1350mV   0mA     1350mV  1350mV
vtt_ddr           0  0     0   675mV   0mA      675mV   675mV
vdd                0  1     0  3300mV   0mA     3300mV  3300mV
vref               1  1     0  2500mV   0mA     2500mV  2500mV
48003000.adc      0  0     0    0mV      0mA      0mV      0mV
v3v3               0  5     0  3300mV   0mA     3300mV  3300mV
0-004a            0  0     0    0mV      0mA      0mV      0mV
58007000.sdmmc    0  0     0  3300mV   0mA     3300mV  3300mV
58005000.sdmmc    0  0     0  3300mV   0mA     3300mV  3300mV
v1v8_audio        0  3     0  1800mV   0mA     1800mV  1800mV
0-004a            0  0     0    0mV      0mA      0mV      0mV
0-004a            0  0     0    0mV      0mA      0mV      0mV
0-004a            0  0     0    0mV      0mA      0mV      0mV
v1v2_hdmi         0  1     0  1200mV   0mA     1200mV  1200mV
0-0039            0  0     0    0mV      0mA      0mV      0mV
v3v3_hdmi         0  1     0  3300mV   0mA     3300mV  3300mV
0-0039            0  0     0    0mV      0mA      0mV      0mV
vdd_usb           2  2     0  3300mV   0mA     3300mV  3300mV
phy-5a006000.usbphyc.1 0  0     0    0mV      0mA      0mV      0mV
phy-5a006000.usbphyc.0 0  0     0    0mV      0mA      0mV      0mV
vdda              0  0     0  2900mV   0mA     2900mV  2900mV
bst_out           0  2     0  5000mV   0mA      0mV      0mV
vbus_otg          0  0     0  5000mV   0mA      0mV      0mV
vbus_sw           0  0     0  5000mV   0mA      0mV      0mV
reg11             1  1     0  1100mV   0mA     1100mV  1100mV
5a006000.usbphyc  0  0     0    0mV      0mA      0mV      0mV
reg18             2  2     0  1800mV   0mA     1800mV  1800mV
5a006000.usbphyc  0  0     0    0mV      0mA      0mV      0mV
5a000000.dsi      0  0     0    0mV      0mA      0mV      0mV
usb33             1  1     0  3300mV   0mA     3300mV  3300mV
49000000.usb-otg  0  0     0    0mV      0mA      0mV      0mV
vref_ddr          0  0     0   675mV   0mA      0mV      0mV
```

- To list the pin control settings:



```

Board $> cat /sys/kernel/debug/pinctrl/soc\:\:pin-controller@50002000/pinconf-pins
Pin config settings per pin
Format: pin (name): configs
pin 0 (PA0): analog
pin 1 (PA1): alternate 11 (ETH1_GMII_RX_CLK ETH1_MII_RX_CLK ETH1_RGMII_RX_CLK
ETH1_RMII_REF_CLK) - push pull - floating - low speed
pin 2 (PA2): alternate 11 (ETH1_MDIO) - push pull - floating - very high speed
pin 3 (PA3): alternate 14 (LCD_B5) - push pull - floating - high speed
pin 4 (PA4): analog
pin 5 (PA5): analog
pin 6 (PA6): analog
pin 7 (PA7): alternate 11 (ETH1_GMII_RX_DV ETH1_MII_RX_DV ETH1_RGMII_RX_CTL
ETH1_RMII_CRS_DV) - push pull - floating - low speed
pin 8 (PA8): analog
pin 9 (PA9): analog
pin 10 (PA10): output - high - push pull - floating - low speed
pin 11 (PA11): analog
pin 12 (PA12): analog
pin 13 (PA13): output - low - open drain - floating - very high speed
pin 14 (PA14): input - high - floating
pin 15 (PA15): analog
pin 16 (PB0): alternate 11 (ETH1_GMII_RXD2 ETH1_MII_RXD2 ETH1_RGMII_RXD2) - push pull -
floating - low speed
pin 17 (PB1): alternate 11 (ETH1_GMII_RXD3 ETH1_MII_RXD3 ETH1_RGMII_RXD3) - push pull -
floating - low speed
pin 18 (PB2): alternate 8 (UART4_RX) - push pull - floating - low speed
pin 19 (PB3): alternate 9 (SDMMC2_D2) - push pull - pull up - very high speed
pin 20 (PB4): alternate 9 (SDMMC2_D3) - push pull - pull up - very high speed
pin 21 (PB5): analog
pin 22 (PB6): alternate 5 (CEC) - open drain - floating - low speed
pin 23 (PB7): analog
pin 24 (PB8): alternate 14 (LCD_B6) - push pull - floating - high speed
pin 25 (PB9): analog
pin 26 (PB10): analog
pin 27 (PB11): alternate 11 (ETH1_GMII_TX_EN ETH1_MII_TX_EN ETH1_RGMII_TX_CTL ETH1_RMII_TX
_EN) - push pull - floating - very high speed
pin 28 (PB12): analog
pin 29 (PB13): analog
pin 30 (PB14): alternate 9 (SDMMC2_D0) - push pull - pull up - very high speed
pin 31 (PB15): alternate 9 (SDMMC2_D1) - push pull - pull up - very high speed
pin 32 (PC0): alternate 14 (LCD_R5) - push pull - floating - high speed
pin 33 (PC1): alternate 11 (ETH1_MDC) - push pull - floating - very high speed
pin 34 (PC2): alternate 11 (ETH1_GMII_TXD2 ETH1_MII_TXD2 ETH1_RGMII_TXD2) - push pull -
floating - very high speed
pin 35 (PC3): analog
pin 36 (PC4): alternate 11 (ETH1_GMII_RXD0 ETH1_MII_RXD0 ETH1_RGMII_RXD0 ETH1_RMII_RXD0)
- push pull - floating - low speed
pin 37 (PC5): alternate 11 (ETH1_GMII_RXD1 ETH1_MII_RXD1 ETH1_RGMII_RXD1 ETH1_RMII_RXD1)
- push pull - floating - low speed
pin 38 (PC6): analog
pin 39 (PC7): analog
pin 40 (PC8): analog
pin 41 (PC9): analog
pin 42 (PC10): analog
pin 43 (PC11): analog
pin 44 (PC12): analog
pin 45 (PC13): analog
pin 46 (PC14): analog
pin 47 (PC15): analog
pin 48 (PD0): analog
pin 49 (PD1): analog
pin 50 (PD2): analog
pin 51 (PD3): alternate 7 (USART2_CTS USART2_NSS) - push pull - floating - low speed
pin 52 (PD4): alternate 7 (USART2_RTS USART2_DE) - push pull - floating - very high speed
pin 53 (PD5): alternate 7 (USART2_TX) - push pull - floating - very high speed

```



```

pin 54 (PD6): alternate 7 (USART2_RX) - push pull - floating - low speed
pin 55 (PD7): analog
pin 56 (PD8): alternate 14 (LCD_B7) - push pull - floating - high speed
pin 57 (PD9): alternate 14 (LCD_B0) - push pull - floating - high speed
pin 58 (PD10): alternate 14 (LCD_B3) - push pull - floating - high speed
pin 59 (PD11): output - low - push pull - floating - low speed
pin 60 (PD12): alternate 5 (I2C1_SCL) - open drain - floating - low speed
pin 61 (PD13): analog
pin 62 (PD14): analog
pin 63 (PD15): analog
pin 64 (PE0): alternate 10 (SAI2_MCLK_A) - push pull - floating - high speed
pin 65 (PE1): analog
pin 66 (PE2): alternate 11 (ETH1_GMII_TXD3 ETH1_MII_TXD3 ETH1_RGMII_TXD3) - push pull -
floating - very high speed
pin 67 (PE3): alternate 9 (SDMMC2_CK) - push pull - pull up - very high speed
pin 68 (PE4): output - high - push pull - floating - low speed
pin 69 (PE5): alternate 14 (LCD_G0) - push pull - floating - high speed
pin 70 (PE6): alternate 14 (LCD_G1) - push pull - floating - high speed
pin 71 (PE7): analog
pin 72 (PE8): analog
pin 73 (PE9): analog
pin 74 (PE10): analog
pin 75 (PE11): analog
pin 76 (PE12): analog
pin 77 (PE13): analog
pin 78 (PE14): analog
pin 79 (PE15): alternate 14 (LCD_R7) - push pull - floating - high speed
pin 80 (PF0): analog
pin 81 (PF1): analog
pin 82 (PF2): input - high - floating
pin 83 (PF3): analog
pin 84 (PF4): analog
pin 85 (PF5): analog
pin 86 (PF6): analog
pin 87 (PF7): analog
pin 88 (PF8): analog
pin 89 (PF9): analog
pin 90 (PF10): alternate 14 (LCD_DE) - push pull - floating - high speed
pin 91 (PF11): alternate 10 (SAI2_SD_B) - push pull - floating - low speed
pin 92 (PF12): analog
pin 93 (PF13): analog
pin 94 (PF14): analog
pin 95 (PF15): alternate 5 (I2C1_SDA) - open drain - floating - low speed
pin 96 (PG0): analog
pin 97 (PG1): input - high - floating
pin 98 (PG2): analog
pin 99 (PG3): analog
pin 100 (PG4): alternate 11 (ETH1_GMII_GTX_CLK ETH1_RGMII_GTX_CLK) - push pull - floating
- very high speed
pin 101 (PG5): alternate 11 (ETH1_GMII_CLK125 ETH1_RGMII_CLK125) - push pull - floating -
very high speed
pin 102 (PG6): alternate 10 (SDMMC2_CMD) - push pull - pull up - very high speed
pin 103 (PG7): alternate 14 (LCD_CLK) - push pull - floating - high speed
pin 104 (PG8): analog
pin 105 (PG9): output - high - push pull - floating - low speed
pin 106 (PG10): alternate 14 (LCD_B2) - push pull - floating - high speed
pin 107 (PG11): alternate 6 (UART4_TX) - push pull - floating - low speed
pin 108 (PG12): alternate 14 (LCD_B1) - push pull - floating - high speed
pin 109 (PG13): alternate 11 (ETH1_GMII_TXD0 ETH1_MII_TXD0 ETH1_RGMII_TXD0
ETH1_RMII_TXD0) - push pull - floating - very high speed
pin 110 (PG14): alternate 11 (ETH1_GMII_TXD1 ETH1_MII_TXD1 ETH1_RGMII_TXD1
ETH1_RMII_TXD1) - push pull - floating - very high speed
pin 111 (PG15): analog
pin 112 (PH0): analog
pin 113 (PH1): analog
pin 114 (PH2): alternate 14 (LCD_R0) - push pull - floating - high speed
pin 115 (PH3): alternate 14 (LCD_R1) - push pull - floating - high speed

```



```

pin 116 (PH4): output - high - push pull - floating - low speed
pin 117 (PH5): analog
pin 118 (PH6): analog
pin 119 (PH7): analog
pin 120 (PH8): alternate 14 (LCD_R2) - push pull - floating - high speed
pin 121 (PH9): alternate 14 (LCD_R3) - push pull - floating - high speed
pin 122 (PH10): alternate 14 (LCD_R4) - push pull - floating - high speed
pin 123 (PH11): analog
pin 124 (PH12): alternate 14 (LCD_R6) - push pull - floating - high speed
pin 125 (PH13): alternate 14 (LCD_G2) - push pull - floating - high speed
pin 126 (PH14): alternate 14 (LCD_G3) - push pull - floating - high speed
pin 127 (PH15): alternate 14 (LCD_G4) - push pull - floating - high speed
pin 128 (PI0): alternate 14 (LCD_G5) - push pull - floating - high speed
pin 129 (PI1): alternate 14 (LCD_G6) - push pull - floating - high speed
pin 130 (PI2): alternate 14 (LCD_G7) - push pull - floating - high speed
pin 131 (PI3): analog
pin 132 (PI4): alternate 14 (LCD_B4) - push pull - floating - high speed
pin 133 (PI5): alternate 10 (SAI2_SCK_A) - push pull - floating - medium speed
pin 134 (PI6): alternate 10 (SAI2_SD_A) - push pull - floating - medium speed
pin 135 (PI7): alternate 10 (SAI2_FS_A) - push pull - floating - medium speed
pin 136 (PI8): analog
pin 137 (PI9): alternate 14 (LCD_VSYNC) - push pull - floating - high speed
pin 138 (PI10): alternate 14 (LCD_HSYNC) - push pull - floating - high speed
pin 139 (PI11): analog

```

- To obtain the GPIO settings:

```

Board $> cat /sys/kernel/debug/gpio
gpiochip0: GPIOs 0-15, parent: platform/soc:pin-controller@50002000, GPIOA:
  gpio-10 (          |reset          ) out lo

gpiochip1: GPIOs 16-31, parent: platform/soc:pin-controller@50002000, GPIOB:

gpiochip2: GPIOs 32-47, parent: platform/soc:pin-controller@50002000, GPIOC:

gpiochip3: GPIOs 48-63, parent: platform/soc:pin-controller@50002000, GPIOD:
  gpio-59 (          |heartbeat      ) out lo

gpiochip4: GPIOs 64-79, parent: platform/soc:pin-controller@50002000, GPIOE:
  gpio-68 (          |reset          ) out hi

gpiochip5: GPIOs 80-95, parent: platform/soc:pin-controller@50002000, GPIOF:

gpiochip6: GPIOs 96-111, parent: platform/soc:pin-controller@50002000, GPIOG:
  gpio-105 (         |reset          ) out hi

gpiochip7: GPIOs 112-127, parent: platform/soc:pin-controller@50002000, GPIOH:
  gpio-116 (         |reset          ) out hi

gpiochip8: GPIOs 128-143, parent: platform/soc:pin-controller@50002000, GPIOI:

gpiochip9: GPIOs 400-415, parent: platform/soc:pin-controller-z@54004000, GPIOZ:

```



4 To go further

4.1 Regmap (register map) cache

Debugfs contains a register cache (mirror) for drivers/peripherals based on regmap API^[3].

Regmap is a register map abstraction API of the Linux kernel. It is mainly used for serial bus device drivers (I2C and SPI), but can also be used for internal peripheral drivers.

Many of these drivers contain some very similar code for accessing connected-hardware device registers.

The regmap kernel API proposes a solution which factors out this code from the drivers, saving code and making it much easier to share infrastructure.

- Path of the regmap register cache in debugfs

```
Board $> cd /sys/kernel/debug/regmap
```

- List of devices for which drivers are based on regmap API:

```
Board $> ls -la .
total 0
drwxr-xr-x 18 root root 0 Jan 1 1970 .
drwx----- 32 root root 0 Jan 1 1970 ..
drwxr-xr-x 2 root root 0 Jan 1 1970 0-001b
drwxr-xr-x 2 root root 0 Jan 1 1970 0-0042
drwxr-xr-x 2 root root 0 Jan 1 1970 2-0033
drwxr-xr-x 2 root root 0 Jan 1 1970 40004000.timer
drwxr-xr-x 2 root root 0 Jan 1 1970 4000d000.audio-controller
drwxr-xr-x 2 root root 0 Nov 30 12:19 40016000.cec
drwxr-xr-x 2 root root 0 Jan 1 1970 40017000.dac
drwxr-xr-x 2 root root 0 Jan 1 1970 4400b004.audio-controller
drwxr-xr-x 2 root root 0 Jan 1 1970 4400b024.audio-controller
drwxr-xr-x 2 root root 0 Jan 1 1970 4400d000.dfsdm
drwxr-xr-x 2 root root 0 Jan 1 1970 50027004.audio-controller
drwxr-xr-x 2 root root 0 Jan 1 1970 dummy-interrupt-controller@5000d000
drwxr-xr-x 2 root root 0 Jan 1 1970 dummy-pwr@50001000
drwxr-xr-x 2 root root 0 Jan 1 1970 dummy-rcc@50000000
drwxr-xr-x 2 root root 0 Jan 1 1970 dummy-syscon@50020000
drwxr-xr-x 2 root root 0 Jan 1 1970 dummy-tamp@5c00a000
```

- Check for the corresponding driver: example for **2-0033** regmap entry

```
Board $> cat 2-0033/name
stpmic1
```

Then, on the STM32MPU device tree files, you can check for the device, and the corresponding driver

```
pmic: stpmic@33 {
    compatible = "st,stpmic1";
    reg = <0x33>;
    interrupts-extended = <&exti_pwr 55 IRQ_TYPE_EDGE_FALLING>;
    interrupt-controller;
    #interrupt-cells = <2>;
    status = "okay";
```




- Regmap entries stating by **dummy-**

This kind of entry is set when there are no associated /dev entries (devtmpfs).

To find the corresponding node in the device tree, you can look for the suffix name after **dummy-**:

Example for **dummy-interrupt-controller@5000d000**

```
exti: interrupt-controller@5000d000 {
    compatible = "st,stm32mp1-exti", "syscon";
    interrupt-controller;
    #interrupt-cells = <2>;
    reg = <0x5000d000 0x400>;
};
```

- List of registers: example for **2-0033** regmap entry

```
Board $> cat 2-0033/registers
01: 10
02: 00
03: 00
04: 00
05: 60
06: 10
10: 04
11: 00
12: 00
13: 00
14: 00
15: c0
16: 00
17: 00
18: 04
19: 00
1a: 00
1b: 00
1c: 00
1d: 0f
1e: 04
20: 61
21: 79
22: d9
23: d9
24: 01
25: 51
26: 4c
27: 7d
28: 01
29: 51
2a: 25
30: 61
31: 50
32: d9
33: d9
34: 00
35: 24
36: 24
37: 24
38: 01
39: 51
3a: 04
40: 35
```




```
50: 00
51: 00
52: 00
53: 80
60: 00
61: 00
62: 00
63: 00
70: 00
71: 00
72: 00
73: 00
80: fc
81: 8f
82: c4
83: cf
90: fc
91: 8f
92: c4
93: cf
a0: 03
a1: 70
a2: 3b
a3: 00
b0: 00
b1: 00
b2: 00
b3: 80
```

The first column represents the register address, and the second column the register value.

For register definitions, please refer to the device specification.



5 References

- Documentation/filesystems/debugfs.txt
- <https://opensourceforu.com/2010/10/debugging-linux-kernel-with-debugfs>
- https://elinux.org/images/a/a3/Regmap-_The_Power_of_Subsystems_and_Abstractions.pdf

- Useful external links

Document link	Document Type	Description
Wikipedia debugfs	Standard	Wikipedia
Guide to debugfs	Standard	Guide
regmap: Reducing the Redundancy in Linux Code	Article	Guide

Random Access Memory (Early computer memories generally had serial 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-access semiconductor memories.)

Process File System (See <https://en.wikipedia.org/wiki/Procfs> for more details)

System File System (See <https://en.wikipedia.org/wiki/Sysfs> for more details)

Debug File System (See <https://en.wikipedia.org/wiki/Debugfs> for more details)

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

Receive

Consumer Electronics Control (HDMI standard)

Transmit

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

Serial clock line

Secure digital

Serial DATA line

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.)

Application programming interface

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

Serial Peripheral Interface

Stable: 31.03.2021 - 08:47 / Revision: 26.03.2021 - 08:44

A quality version of this page, approved on 31 March 2021, was based off this revision.



Contents

1 Linux configuration genericity	20
2 Menuconfig and Developer Package	22
3 Menuconfig and Distribution Package	24
4 References	25

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/5.10.10-rc0/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)

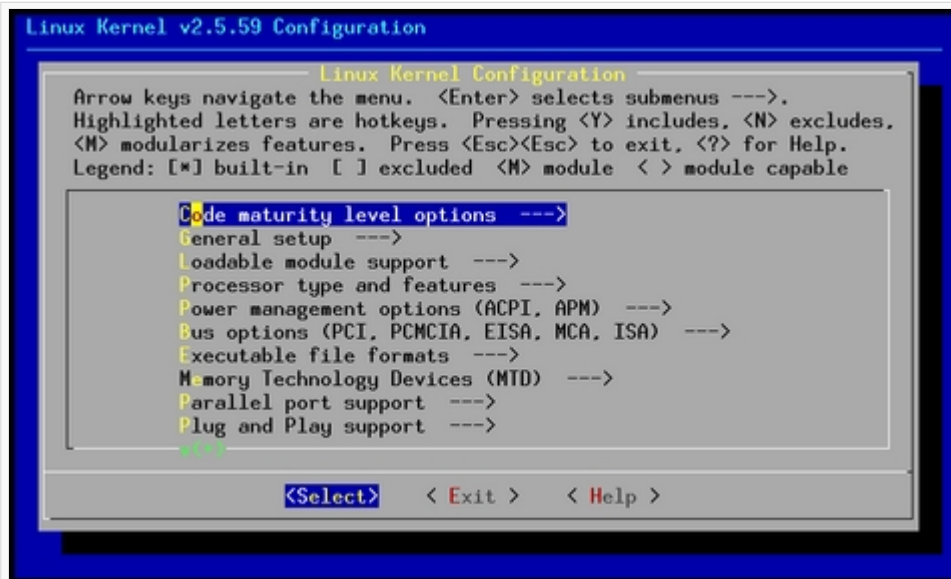
```
[greg@shamp linux-2.5]$ make config
make[1]: `scripts/kconfig/conf' is up to date.
./scripts/kconfig/conf arch/i386/Kconfig
#
# using defaults found in .config
#
*
* Linux Kernel Configuration
*
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers (EXPERIMENTAL) [Y/n/?]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

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^{\[1\]}](#)
^[1] also explains how to "navigate" within the configuration menu, and highlights main key strokes.

configurator :

- or a graphical kernel



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.

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
```

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

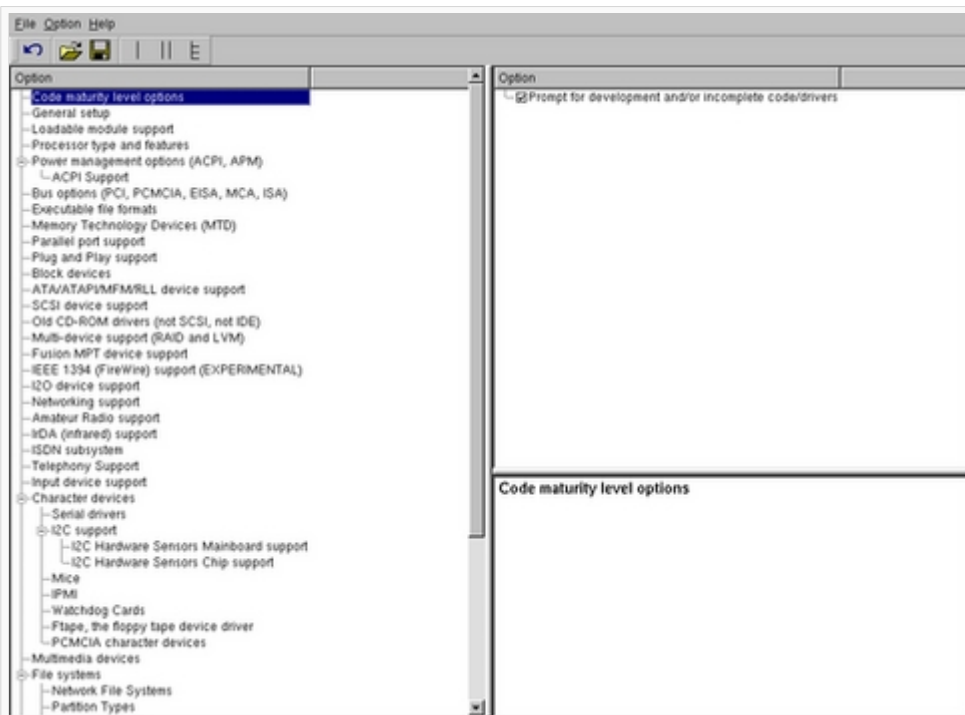


Figure 3. The Qt-Based make xconfig



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/
```



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
```



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.)

Stable: 31.01.2020 - 13:23 / Revision: 31.01.2020 - 13:16

A quality version of this page, approved on 31 January 2020, was based off this revision.

Contents

1 Introduction	26
2 procfs (/proc) - Kernel and process information	27
2.1 Some procfs useful entries	27
3 sysfs (/sys) - System filesystem	29
3.1 Top Level sysfs directory layout	29
3.2 Zoom to debugfs (/sys/kernel/debug)	31
3.3 Zoom to configfs (/sys/kernel/config)	31
3.4 Zoom to tracefs (/sys/kernel/tracing)	31
4 Information about temporary filesystem	32
4.1 tmpfs	32
4.2 devtmpfs	32
5 References	34



1 Introduction

When running, Linux[®] operating system creates and populates some filesystems which are not present on the rootfs filesystem of the Linux distribution image:

- **pseudo filesystems:** **sysfs** (/sys), **procfs** (/proc), **debugfs** (/sys/kernel/debug), **configfs** (/sys/kernel/config), **tracefs** (/sys/kernel/tracing)
- **temporary filesystems:** **tmpfs** (/dev/shm, /run, /sys/fs/cgroup, /tmp/, /var/volatile, /run/user/<id>), **devtmpfs** (/dev)

Pseudo filesystems contain many informations, configurations and logs about the current running kernel. Informations from those pseudo filesystems are very helpful and any debugging session should start by browsing them.

These both filesystem groups are part of the [File Hierarchy Standard \(FHS\)](#) for the Linux operating system.

As they are placed in volatile memory, they are only available at run time, and they disappear at shutdown.



2 procfs (/proc) - Kernel and process information

Procfs^[1] is enabled and ready to be used in all STM32MPU Embedded Software distribution, via the Linux® kernel configuration **CONFIG_PROC_FS**, set to yes by default.

```
Symbol: PROC_FS
Location:
  File systems --->
    Pseudo filesystems -->
      [*] /proc file system support
```

Please refer to [Menuconfig](#) or [how to configure kernel](#) article to get instructions for modifying the configuration and recompiling the Linux kernel image in the Distribution Package context.

Procfs is sometimes referred to as a process information pseudo-file system. It does not contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc).

For this reason it can be seen as a control and information center for the kernel. In fact, quite a lot of system utilities are simply calls to files in this directory.

For example, 'lsmod' is the same as 'cat /proc/modules' while 'lspci' is a synonym for 'cat /proc/pci'. By altering files located in this directory you can even read/change kernel parameters (sysctl) while the system is running.

Procfs is explain in details in [The Linux Documentation Project](#)^[2], or [Wikipedia](#)^[3].

2.1 Some procfs useful entries

Name	Description
/proc/<P ID>/	directory containing information about the kernel process <PID>
/proc /cmdline	gives the boot options passed to the kernel
/proc /cpuinfo	provides information about the processor: its type, make, model, and performance
/proc /devices	lists all of the devices (divided into the "block" and "character" categories) available on the currently running system, giving also the major part of the /dev name too
/proc /device- tree	contains devices tree information for all nodes organized as defined in the device-tree source file(s).
/proc /interrupts	shows which interrupts are in use, and how many of each there have been.
/proc	



Name	Description
/iomem	gives information about memory mapping
/proc /kallsym s	contains the dynamic kernel symbol table
/proc /kmsg	holds messages output by the kernel.
/proc /meminf o	contains a summary of how the kernel is managing its memory (both physical and swap).
/proc /misc	Miscellaneous pieces of information, lists also the misc features devices
/proc /module s	one of the most important files in /proc; contains a list of the kernel modules currently loaded. It gives some indication of dependencies.



3 sysfs (/sys) - System filesystem

Sysfs^[4] is a RAM-based filesystem initially based on ramfs. It provides a means to export kernel data structures, their attributes, and the linkages between them, to user space.

Sysfs is enabled and ready to be used in all STM32MPU Embedded Software distribution, via the Linux[®] kernel configuration **CONFIG_SYSFS**, set to yes by default.

```
Symbol: SYSFS
Location:
  File systems --->
  Pseudo filesystems -->
    [*] sysfs file system support
```

Please refer to [Menuconfig](#) or [how to configure kernel](#) article to get instructions for modifying the configuration and recompiling the Linux kernel image in the Distribution Package context.

Useful information also given in [How to access information in sysfs](#) article.

3.1 Top Level sysfs directory layout

Sysfs directory arrangement exposes the relationship of kernel data structures.

/sys has a sub-hierarchy file system:

su b- Dir ect ory	Description
/s ys /bl oc k/	Contains one symbolic link for each block device that has been discovered on the system. The symbolic links point to corresponding directories under /sys/devices.
/s ys /b us /	Contains one subdirectory for each of the bus types in the kernel. Each bus's directory contains two subdirectories: ./devices/ ./drivers/
/s ys /cl as s/	Contains a single layer of further subdirectories for each of the device classes that have been registered on the system (e.g., terminals, network devices, block devices, graphics devices, sound devices, and so on). Inside each of these subdirectories are symbolic links for each of the devices in this class. These symbolic links refer to entries in the /sys/devices directory



su b- Dir ect ory	Description
/s ys /cl as s /n et /	Each of the entries in this directory is a symbolic link representing one of the real or virtual networking devices that are visible in the network namespace of the process that is accessing the directory. Each of these symbolic links refers to entries in the /sys/devices directory
/s ys /d ev /	<p>This directory contains two subdirectories block/ and char/, corresponding, respectively, to the block and character devices on the system. Inside each of these subdirectories are symbolic links with names of the form major-ID:minor-ID, where the ID values correspond to the major and minor ID of a specific device.</p> <p>Each symbolic link points to the sysfs directory for a device. The symbolic links inside /sys/dev thus provide an easy way to look up the sysfs interface using the device IDs returned by a call to stat tool (or similar)</p>
/s ys /d ev ic es /	Contains a filesystem representation of the kernel device tree, which is a hierarchy of device structures within the kernel
/s ys /fi r m w ar e/	Contains interfaces for viewing and manipulating firmware-specific objects and attributes
/s ys /fs /	Contains subdirectories for some filesystems. A filesystem will have a subdirectory here only if it chose to explicitly create the subdirectory
/s ys /k	



sub-Directory	Description
kernel/	Contains various files and subdirectories that provide information about the running kernel
/sys/module/	Contains one subdirectory for each module that is loaded into the kernel. The name of each directory is the name of the module

3.2 Zoom to debugfs (/sys/kernel/debug)

Please refer to [debugfs](#) article.

3.3 Zoom to configfs (/sys/kernel/config)

Please refer to [configfs](#) article.

3.4 Zoom to tracefs (/sys/kernel/tracing)

Tracefs is used with the Linux kernel tracing framework.

Example of usage is given in [Ftrace](#) article.

- Command to mount tracefs:

```
Board $> mount -t tracefs nodev /sys/kernel/tracing
```

To find out which tracers are available, simply read `available_tracers` file in the tracing directory:

```
Board $> cat /sys/kernel/tracing/available_tracers
function_graph function nop
```

More tracers can be added by kernel build configurations. Please refer to [Ftrace#More_tracers](#) paragraph.



4 Information about temporary filesystem

4.1 tmpfs

Tmpfs^[5] is a file system which keeps all files in virtual memory.

It is enabled and ready to be used in all STM32MPU Embedded Software distribution, via the Linux[®] kernel configuration **CONFIG_TMPFS**, set to yes by default.

```
Symbol: TMPFS
Location:
  File systems --->
    Pseudo filesystems -->
      [*] Tmpfs virtual memory file system support (former shm fs)
```

Please refer to [Menuconfig](#) or [how to configure kernel](#) article to get instructions for modifying the configuration and recompiling the Linux kernel image in the Distribution Package context.

Everything in tmpfs is temporary in the sense that no files will be created on your hard drive. If you unmount a tmpfs instance, everything stored therein is lost.

On the board target, you can check for the directory path mount with the tmpfs:

```
Board $> mount | grep tmpfs
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
tmpfs on /tmp type tmpfs (rw,nosuid,nodev)
tmpfs on /var/volatile type tmpfs (rw,relatime)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=43812k,mode=700)
```

For all details you can refer to the Linux documentation about tmpfs^[5].

4.2 devtmpfs

Devtmpfs is enabled and ready to be used in all STM32MPU Embedded Software distribution, via the Linux[®] kernel configuration **CONFIG_DEVTMPFS** and **CONFIG_DEVTMPFS_MOUNT**, set to yes by default.



```

Symbol: DEVTMPFS
Location:
  Device Drivers --->
    Generic Driver Options -->
      [*] Maintain a devtmpfs filesystem to mount at /dev

Symbol: DEVTMPFS_MOUNT
Location:
  Device Drivers --->
    Generic Driver Options -->
      [*] Maintain a devtmpfs filesystem to mount at /dev
      [*] Automount devtmpfs at /dev, after the kernel mounted the rootfs

```

- **/dev** - special or device files

Devtmpfs is mounted on /dev which is the location of special or device files. Many of these are generated at boot time or even on the fly.

It is a very interesting directory that highlights one important aspect of the Linux filesystem: **everything is a file or a directory**.

Look through this directory and you can see device file system entries which represent the various partitions on the first master drive of the system:

For example:

- mmcblk0p<id> (microSD Card),
- mmcblk1p<id> (eMMC),
- sda<id>,
- sdb<id> (NAND or USB Key),
- ttySTM<id> (tty Serial link),
- etc...

These entries can be both read from and written to.

Take /dev/ttyUSB0, for instance. This file represents the USB Serial port. Sending data to and reading from /dev/ttyUSB0 will allow you to communicate with host PC through the minicom application (or equivalent).

/dev is very helpful, more info could be found in the Linux Documentation Project^[6].

5 References

- Documentation/filesystems/proc.txt
- <http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html>
- <https://en.wikipedia.org/wiki/Procfs>
- Documentation/filesystems/sysfs.txt
- 5.05.1 Documentation/filesystems/tmpfs.txt
- <http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/dev.html>

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

System File System (See <https://en.wikipedia.org/wiki/Sysfs> for more details)

Process File System (See <https://en.wikipedia.org/wiki/Procfs> for more details)

Debug File System (See <https://en.wikipedia.org/wiki/Debugfs> for more details)

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

Random Access Memory (Early computer memories generally had serial 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-access semiconductor memories.)

Read Only

former spelling for eMMC ('e' in italic)

Stable: 18.03.2021 - 15:06 / Revision: 18.03.2021 - 15:05

A quality version of this page, approved on 18 March 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).

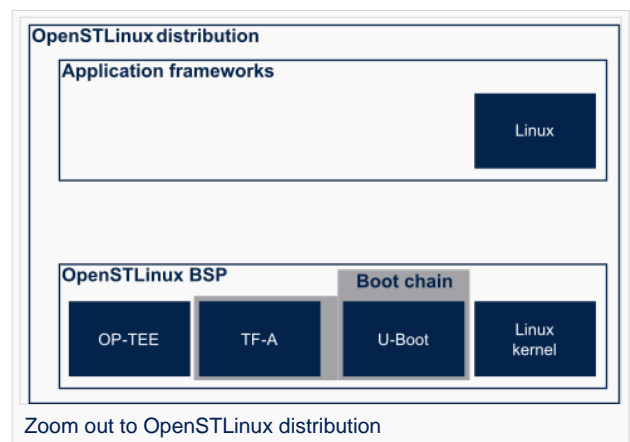
The components are grouped per **functional domains**.

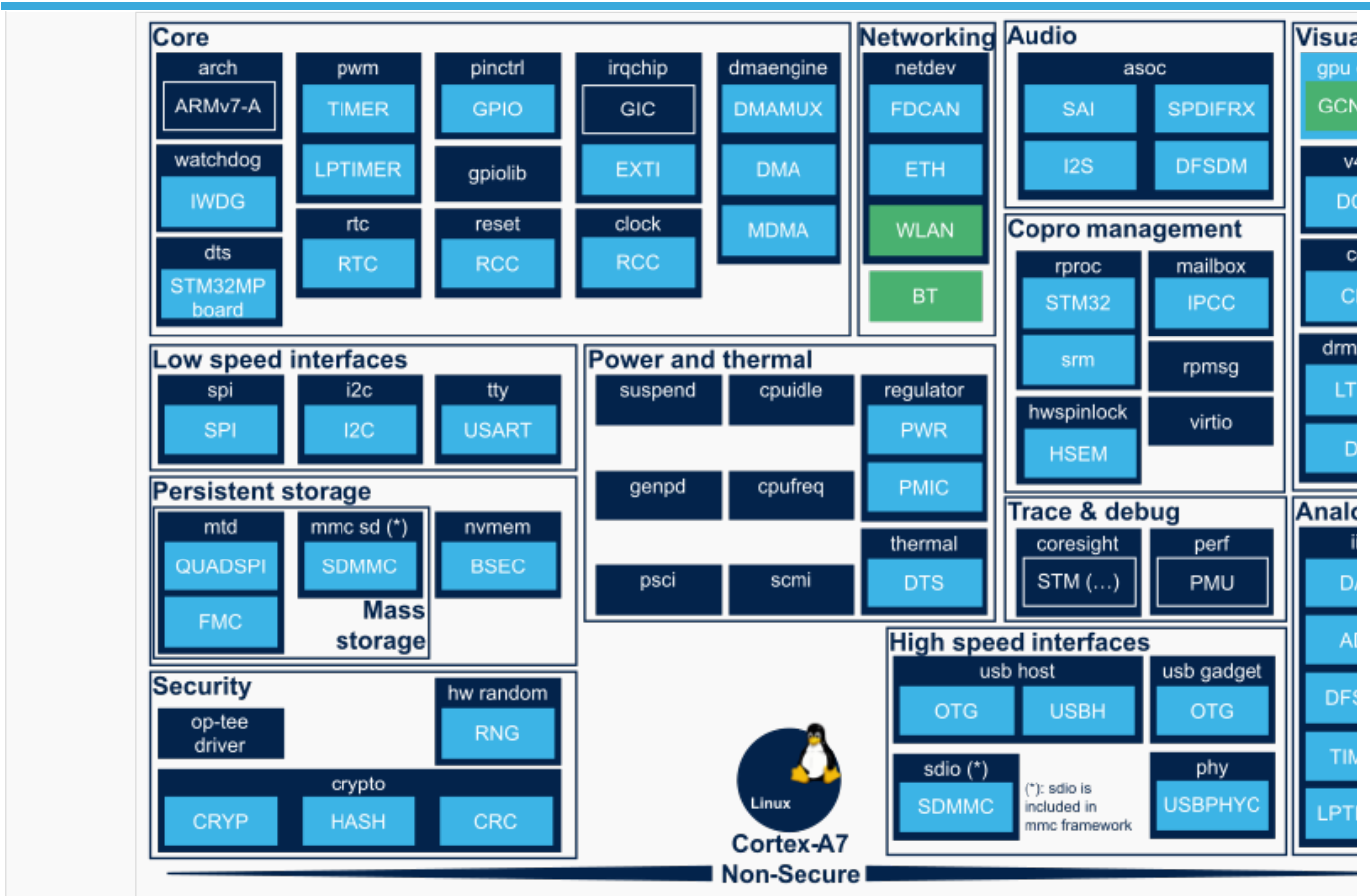
Each **Linux framework** is further described in [Linux operating system category articles](#).

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