



DAC Linux driver



Contents



A quality version of this page, approved on *16 January 2020*, was based off this revision.

Contents

1 Article purpose	4
2 Short Description	5
3 Configuration	6
3.1 Kernel configuration	6
3.2 Device tree	6
4 How to use	7
5 How to trace and debug	8
6 Source code location	9
7 References	10



1 Article purpose

This article introduces the Linux[®] driver for the DAC^[1] internal peripheral:

- Which DAC features are supported by the driver
- How to configure, use and debug the driver
- What is the driver structure, and where the source code can be found.



2 Short Description

The DAC Linux[®] driver (kernel space) is based on the IIO framework.

It implements the **IIO direct mode**, to perform single conversions independently on each channel.



3 Configuration

3.1 Kernel configuration

Activate the DAC^[1]Linux[®] driver in the kernel configuration using the Linux Menuconfig tool: Menuconfig or how to configure kernel (enable CONFIG_STM32_DAC).

```
Device Drivers --->
  <*> Industrial I/O support --->
    Digital to analog converters --->
      <*> STMicroelectronics STM32 DAC
```

3.2 Device tree

Refer to the [DAC device tree configuration](#) article when configuring the DAC Linux kernel driver.



4 How to use

In "IIO direct mode", conversions can be done directly via sysfs. See [How to do a simple DAC conversion using the sysfs interface](#).



5 How to trace and debug

Refer to [How to trace with dynamic debug](#) for how to enable debug logs in the driver and in the Framework.

Refer to [How to debug with debugfs](#) for how to access the DAC registers.

The DAC has system wide dependencies towards other key resources:

- **runtime power management** can be disabled, for example it may be forced **on** via *power/control* sysfs entry:

```
Board $> cd /sys/devices/platform/soc/40017000.dac/40017000.dac\:dac@1/
Board $> cat power/autosuspend_delay_ms
2000
Board $> cat power/control
auto # kernel is allowed to automatically suspend the
ADC device after autosuspend_delay_ms
Board $> echo on > power/control # force the kernel to resume the DAC device (e.
g. keep clocks and regulators enabled)
```



It might be useful to disable runtime power management, in order to dump registers by any means or to check clock and regulator usage (see example below).

- **clock**^[2] usage can be verified by reading *clk_summary*:

```
Board $> cat /sys/kernel/debug/clk/clk_summary | grep dac
dac12_k          0 0 0 32000 0 0
                dac12 1 2 0 98303955 0 0
```

- **regulator**^[3] tree and usage usage can be verified (e.g. use count, open count and regulator reference voltage) as follows:

```
Board $> cat /sys/kernel/debug/regulator/regulator_summary
regulator      use open bypass voltage current      min      max
-----
v3v3           4 5 0 3300mV 0mA 3300mV 3300mV
vdda           1 2 0 2900mV 0mA 2900mV 2900mV
40017000.dac  0 0 0 0mV 0mA 0mV 0mV
48003000.adc  0 0 0 0mV 0mA 0mV 0mV
```

- **pinctrl**^[4] usage can be verified by reading *pinmux-pins*:

```
Board $> cd /sys/kernel/debug/pinctrl/soc\:pin-controller@50002000/
Board $> cat pinmux-pins | grep dac
pin 4 (PA4): device 40017000.dac function analog group PA4
pin 5 (PA5): device 40017000.dac function analog group PA5 # check pins are assigned to
DAC and configured as "analog"
```




6 Source code location

The DAC source code is composed of:

- `stm32-dac-core` driver to handle common resources such as `clock` or `regulator` used as reference voltage and common registers.
- `stm32-dac` driver to handle the resources available for each DAC such as channel configuration or output buffer handling (power-down mode).



7 References

- 1.01.1 DAC internal peripheral
- Clock overview
- Regulator overview
- Pinctrl overview

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

Digital-to-analog converter (Electronic circuit that converts a binary number into a continuously varying value.)

Industrial I/O Linux[®] subsystem

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

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.