One Technology Way · P.O. Box 9106 · Norwood, MA 02062-9106 · Tel: 781.329.4700 · Fax: 781.461.3113 · www.analog.com

AD5446 IIO DAC Linux Driver

Supported Devices

- AD5300
- AD5301
- AD5310
- AD5311
- AD5320
- AD5321
- AD5444
- AD5446
- AD5450
- AD5451
- AD5452
- AD5453
- AD5512A
- AD5541A
- AD5542A
- AD5543
- AD5553
- AD5601
- AD5602
- AD5611
- AD5612
- AD5620
- AD5620-1
- AD5620-2
- AD5620-3
- AD5621
- AD5622
- AD5640
- AD5640-1
- AD5640-2
- AD5640-3
- AD5641
- AD5660
- AD5660-1
- AD5660-2
- AD5660-3
- AD5662

Reference Circuits

- CN0009
- CN0050
- CN0052
- CN0053
- CN0055
- CN0063
- CN0064
- CN0079
- CN0082
- CN0143
- CN0151
- CN0156
- CN0169
- CN0179
- CN0181
- CN0202
- CN0203
- CN0204
- CN0348

Evaluation Boards

- EVAL-AD5446SDZ
- EVAL-AD5453SDZ
- EVAL-AD5541ASDZ
- EVAL-AD5542ASDZ
- EVAL-AD5543SDZ
- EVAL-AD5621EBZ
- EVAL-AD5620EBZ
- EVAL-AD5660EBZ
- EVAL-AD5662EBZ

Description

This is a Linux industrial I/O (IIO) subsystem driver, targeting single channel serial interface DACs. The industrial I/O subsystem provides a unified framework for drivers for many different types of converters and sensors using a number of different physical interfaces (i2c, spi, etc). See IIO for more information.

Source Code

Status

Source	Mainlined?
git	Yes

Files

Function	File
driver	drivers/iio/dac/ad5446.c
include	drivers/iio/dac/ad5446.h

Example platform device initialization

Specifying reference voltage via the regulator framework

For chip variants with build in references such as the: ad5620-2500 ad5620-1250 ad5640-2500 ad5640-1250 ad5660-2500 ad5660-1250 specifying the reference voltage via the regulator framework is not mandatory.

Below example specifies a 2.5 Volt reference for the SPI device 3 on SPI-Bus 0. (**spi0.3**)

```
#if defined(CONFIG_REGULATOR_FIXED_VOLTAGE) ||
defined(CONFIG_REGULATOR_FIXED_VOLTAGE_MODULE)
static struct regulator_consumer_supply ad5446_consumer_supplies[] = {
        REGULATOR_SUPPLY("vcc", "spi0.3"),
};

static struct regulator_init_data stamp_avdd_reg_init_data = {
        .constraints        = {
                .name         = "2V5",
                .valid_ops_mask = REGULATOR_CHANGE_STATUS,
        },
        .consumer_supplies = ad5446_consumer_supplies,
```

```
        .num_consumer_supplies = ARRAY_SIZE(ad5446_consumer_supplies),
};

static struct fixed_voltage_config stamp_vdd_pdata = {
        .supply_name        = "board-2V5",
        .microvolts         = 2500000,
        .gpio               = -EINVAL,
        .enabled_at_boot = 0,
        .init_data          = &stamp_avdd_reg_init_data,
};
static struct platform_device brd_voltage_regulator = {
        .name               = "reg-fixed-voltage",
        .id                 = -1,
        .num_resources      = 0,
        .dev                = {
                .platform_data        = &stamp_vdd_pdata,
        },
};
#endif
```

```
static struct platform_device *board_devices[] __initdata = {
#if defined(CONFIG_REGULATOR_FIXED_VOLTAGE) ||
defined(CONFIG_REGULATOR_FIXED_VOLTAGE_MODULE)
        &brd_voltage_regulator
#endif
};
```

```
static int __init board_init(void)
{
        [--snip--]

        platform_add_devices(board_devices, ARRAY_SIZE(board_devices));

        [--snip--]

        return 0;
}
arch_initcall(board_init);
```

Declaring SPI slave devices

Unlike PCI or USB devices, SPI devices are not enumerated at the hardware level. Instead, the software must know which devices are connected on each SPI bus segment, and what slave selects these devices are using. For this reason, the kernel code must instantiate SPI devices explicitly. The most common method is to declare the SPI devices by bus number.

This method is appropriate when the SPI bus is a system bus, as in many embedded systems, wherein

each SPI bus has a number which is known in advance. It is thus possible to pre-declare the SPI devices that inhabit this bus. This is done with an array of struct spi_board_info, which is registered by calling spi_register_board_info().

For more information see: Documentation/spi/spi-summary

21 Oct 2010 15:10 · Michael Hennerich

Depending on the converter IC used, you may need to set the modalias accordingly, matching your part name.

It may also required to adjust max_speed_hz. Please consult the datasheet, for maximum spi clock supported by the device in question.

```
static struct spi_board_info board_spi_board_info[] __initdata = {
#if defined(CONFIG_AD5446) || \
        defined(CONFIG_AD5446_MODULE)
        {
                /* the modalias must be the same as spi device driver name
*/
                .modalias = "ad5446", /* Name of spi_driver for this device
*/
                .max_speed_hz = 1000000,     /* max spi clock (SCK) speed
in HZ */
                .bus_num = 0, /* Framework bus number */
                .chip_select = 3, /* Framework chip select */
                .mode = SPI_MODE_3,
        },
#endif
};
```

```
static int __init board_init(void)
{
        [--snip--]

        spi_register_board_info(board_spi_boar
d_info, ARRAY_SIZE(board_spi_board_info));

        [--snip--]

        return 0;
}
arch_initcall(board_init);
```

Devicetree

Required devicetree properties:

- compatible: Needs to be “adi,” followed by the name of the device. E.g. “adi,ad5446”
- reg: The chipselect number used for the device
- spi-max-frequency: Maximum SPI clock frequency
- vcc-supply: VCC voltage supply regulator

```
       dac_vcc: fixedregulator@1 {
              compatible = "regulator-fixed";
              regulator-name = "fixed-supply";
              regulator-min-microvolt = <2500000>;
              regulator-max-microvolt = <2500000>;
              regulator-boot-on;
       };

       axi_spi_1: spi@42040000 {
              #address-cells = <1>;
              #size-cells = <0>;
              compatible = "xlnx,axi-spi-1.02.a", "xlnx,xps-spi-2.00.a";
               ...
              ad5446@0 {
                     compatible = "adi,ad5446";
                     reg = <0>;
                     spi-max-frequency = <1000000>;
                     spi-cpha;
                     spi-cpol;
                     vcc-supply = <&dac_vcc>;
              };
        };
```

Adding Linux driver support

Configure kernel with “make menuconfig” (alternatively use “make xconfig” or “make qconfig”)

The AD5446 Driver depends on **CONFIG_SPI**

```
Linux Kernel Configuration
        Device Drivers  --->
                <*> Industrial I/O support --->
                        --- Industrial I/O support
                        [--snip--]
                        *** Digital to analog convertors ***
                                [--snip--]
                                <*> Analog Devices AD5446 and similar
```

```
single channel DACs driver
                                [--snip--]
```

Hardware configuration

Driver testing

Each and every IIO device, typically a hardware chip, has a device folder under /sys/bus/iio/devices/iio:deviceX. Where X is the IIO index of the device. Under every of these directory folders reside a set of files, depending on the characteristics and features of the hardware device in question. These files are consistently generalized and documented in the IIO ABI documentation. In order to determine which IIO deviceX corresponds to which hardware device, the user can read the name file /sys/bus/iio/devices/iio:deviceX/name. In case the sequence in which the iio device drivers are loaded/registered is constant, the numbering is constant and may be known in advance.

02 Mar 2011 14:16 · Michael Hennerich

This specifies any shell prompt running on the target

```
root:/> cd /sys/bus/iio/devices/
root:/sys/bus/iio/devices> ls
iio:device0                  iio:device0:buffer0:access0  iio:trigger0
iio:device0:buffer0          iio:device0:buffer0:event0

root:/sys/bus/iio/devices> cd iio:device0

root:/sys/devices/platform/bfin-spi.0/spi0.3/iio:device0> ls -l
-r--r--r--    1 root     root          4096 Jan  2 21:23 dev
-r--r--r--    1 root     root          4096 Jan  2 21:23 name
-rw-r--r--    1 root     root          4096 Jan  2 21:23
out_voltage0_powerdown
--w-------    1 root     root          4096 Jan  2 21:23
out_voltage0_raw
-rw-r--r--    1 root     root          4096 Jan  2 21:23
out_voltage_powerdown_mode
-r--r--r--    1 root     root          4096 Jan  2 21:23
out_voltage_powerdown_mode_available
-r--r--r--    1 root     root          4096 Jan  2 21:23
out_voltage_scale
drwxr-xr-x    2 root     root             0 Jan  2 21:23 power
lrwxrwxrwx    1 root     root             0 Jan  2 21:23 subsystem ->
```

```
../../../../../bus/iio
-rw-r--r--    1 root     root          4096 Jan  2 21:23 uevent
```

Show device name

This specifies any shell prompt running on the target

```
root:/sys/devices/platform/bfin-spi.0/spi0.3/iio:device0> cat name
ad5446
```

Show scale

Description:
scale to be applied to out_voltage0_raw in order to obtain the measured voltage in millivolts.

This specifies any shell prompt running on the target

```
root:/sys/devices/platform/bfin-spi.0/spi0.3/iio:device0> cat
out_voltage_scale
0.152
```

Set channel 0

Description:

/sys/bus/iio/devices/iio:deviceX/out_voltageY_raw

Raw (unscaled, no bias etc.) output voltage for channel Y.

This specifies any shell prompt running on the target

```
root:/sys/devices/platform/bfin-spi.0/spi0.3/iio:device0> echo 1234 >
out_voltage0_raw
```

U = *out_voltage0_raw* * *out_voltage_scale* = 1234 * 0.152 mV = **187.568 *mV***

Output power down modes described below only exist on:

- AD5601
- AD5611
- AD5620
- AD5621
- AD5640
- AD5660

List available power down modes

/sys/bus/iio/devices/iio:deviceX/out_voltage_powerdown_mode_available

Description:
Lists all available output power down modes.

This specifies any shell prompt running on the target

```
root:/sys/devices/platform/bfin-spi.0/spi0.3/iio:device0> cat
out_voltage_powerdown_mode_available
1kohm_to_gnd 100kohm_to_gnd three_state
```

Set power down mode

/sys/bus/iio/devices/iio:deviceX/out_voltage_powerdown_mode

Description:
Specifies the output power down mode. DAC output stage is disconnected from the amplifier and

1kohm_to_gnd	connected to ground via an 1kOhm resistor
100kohm_to_gnd	connected to ground via an 100kOhm resistor
three_state	left floating

For a list of available output power down options read outX_powerdown_mode_available.

This specifies any shell prompt running on the target

```
root:/sys/devices/platform/bfin-spi.0/spi0.3/iio:device0> echo
three_state > out_voltage_powerdown_mode
root:/sys/devices/platform/bfin-spi.0/spi0.3/iio:device0> cat
out_voltage_powerdown_mode
three_state
```

Enable power down mode on output Y

/sys/bus/iio/devices/iio:deviceX/out_voltageY_powerdown

Description:
Writing 1 causes output Y to enter the power down mode specified by the corresponding out_voltageY_powerdown_mode. Clearing returns to normal operation. Y may be suppressed if all outputs are controlled together.

This specifies any shell prompt running on the target

```
root:/sys/devices/platform/bfin-spi.0/spi0.3/iio:device0> echo 1 >
out_voltage0_powerdown
root:/sys/devices/platform/bfin-spi.0/spi0.3/iio:device0> cat
out_voltage0_powerdown
1
root:/sys/devices/platform/bfin-spi.0/spi0.3/iio:device0> echo 0 >
out_voltage0_powerdown
root:/sys/devices/platform/bfin-spi.0/spi0.3/iio:device0> cat
out_voltage0_powerdown
0
```

More Information

- IIO mailing list: linux-iio@vger.kernel.org
- IIO Documentation
- IIO Utils Main Page
- IIO test and visualization demo application
- IIO Command Server
- Pointers and good books

Need Help?

- Analog Devices Linux Device Drivers Help Forum
- Ask a Question

31 Jul 2012 16:53 · Lars-Peter Clausen
02 Mar 2011 14:16 · Michael Hennerich