AD7606 IIO Multi-Channel Simultaneous Sampling ADC Linux Driver

Supported Devices

- AD7606
- AD7606-6
- AD7606-4

Reference Circuits

- CN0148

Evaluation Boards

- EVAL-AD7606EDZ
- EVAL-AD7606-4EDZ
- EVAL-AD7606-6EDZ

Description

This is a Linux industrial I/O (IIO) subsystem driver, targeting multi channel, dual interface serial/parallel interface ADCs. 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/staging/iio/adc/ad7606_core.c
driver	drivers/staging/iio/adc/ad7606_ring.c
driver	drivers/staging/iio/adc/ad7606_spi.c
driver	drivers/staging/iio/adc/ad7606_par.c
include	drivers/staging/iio/adc/ad7606.h

Example platform device initialization

For compile time configuration, it’s common Linux practice to keep board- and application-specific configuration out of the main driver file, instead putting it into the board support file.

For devices on custom boards, as typical of embedded and SoC-(system-on-chip) based hardware, Linux uses platform_data to point to board-specific structures describing devices and how they are connected to the SoC. This can include available ports, chip variants, preferred modes, default initialization, additional pin roles, and so on. This shrinks the board-support packages (BSPs) and minimizes board and application specific #ifdefs in drivers.

21 Oct 2010 15:10 · Michael Hennerich

The reference volatage may vary between boards and models. The platform_data for the device's “struct device” holds this information.

```
/**
 * struct ad7606_platform_data - platform/board specifc information
 * @default_os:     default oversampling value {0, 2, 4, 8, 16, 32, 64}
 * @default_range:  default range +/-{5000, 10000} mVolt
 * @gpio_convst:    number of gpio connected to the CONVST pin
 * @gpio_reset:     gpio connected to the RESET pin, if not used set to -1
 * @gpio_range:     gpio connected to the RANGE pin, if not used set to -1
 * @gpio_os0:       gpio connected to the OS0 pin, if not used set to -1
 * @gpio_os1:       gpio connected to the OS1 pin, if not used set to -1
 * @gpio_os2:       gpio connected to the OS2 pin, if not used set to -1
 * @gpio_frstdata:  gpio connected to the FRSTDAT pin, if not used set to
-1
 * @gpio_stby:      gpio connected to the STBY pin, if not used set to -1
 */
```

file: drivers/staging/iio/adc/ad7606.h

```
scm failed with exit code 1:
file does not exist in git
```

```
static struct ad7606_platform_data ad7606_pdata = {
    .default_os = 0,
    .default_range = 10000,
    .gpio_convst = GPIO_PF9,
    .gpio_reset = GPIO_PF10,
    .gpio_range = GPIO_PH0,
    .gpio_os0 = GPIO_PH1,
    .gpio_os1 = GPIO_PH2,
    .gpio_os2 = GPIO_PH3,
    .gpio_frstdata = -1,
    .gpio_stby = GPIO_PF12,
};
```

Example platform_device initialization / parallel interface

For the memory mapped parallel interface option, the user must specify the physical base address where the AD7606 is mapped into. A system IRQ number for the AD7606 BUSY indicator signal must be specified.

ADI part number	platform_device name
AD7606	ad7606-8
AD7606-6	ad7606-6
AD7606-4	ad7606-4

```
#if defined(CONFIG_AD7606) || defined(CONFIG_AD7606_MODULE)
static struct resource ad7606_resources[] = {
    [0] = {
        .start    = 0x20100000,            /* SDP: AMS1 / CS_B */
        .end    = 0x20100000,
        .flags    = IORESOURCE_MEM,
    },
    [1] = {    /* general IRQ */
        .start    = IRQ_PH6,       /* SDP: GPIO6 */
        .end    = IRQ_PH6,
        .flags    = IORESOURCE_IRQ | IORESOURCE_IRQ_HIGHLEVEL,
    },
};
```

```
static struct platform_device ad7606_device = {
    .name        = "ad7606-8",
    .dev = {
        .platform_data = &ad7606_pdata,
    },
    .num_resources    = ARRAY_SIZE(ad7606_resources),
    .resource    = ad7606_resources,
};
#endif
```

```
static struct platform_device *board_devices[] __initdata = {
#if defined(CONFIG_AD7606) || \
    defined(CONFIG_AD7606_MODULE)
    &ad7606_device,
#endif
};
```

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

    platform_add_devices(board_devices, ARRAY_SIZE(board_devices));

    [--snip--]

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

Example spi_board_info initialization / serial interface

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 accordin gly, 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.

ADI part number	spi_board_info modalias
AD7606	ad7606-8
AD7606-6	ad7606-6
AD7606-4	ad7606-4

```
static struct spi_board_info board_spi_board_info[] __initdata = {
#if defined(CONFIG_AD7606) || \
    defined(CONFIG_AD7606_MODULE)
    {
        /* the modalias must be the same as spi device driver name */
        .modalias = "ad7606-8", /* Name of spi_driver for this device */
        .max_speed_hz = 10000000,     /* max spi clock (SCK) speed in HZ */
        .bus_num = 0, /* Framework bus number */
        .chip_select = 3, /* Framework chip select */
        .platform_data = &ad7606_pdata,
        .controller_data = &ad7606_chip_info, /* Blackfin only */
        .irq = IRQ_PH6,
        .mode = SPI_MODE_3,
    },
#endif
};
```

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

    spi_register_board_info(board_spi_board_info, ARRAY_SIZE(
board_spi_board_info));

    [--snip--]

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

Adding Linux driver support

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

```
Linux Kernel Configuration
    Device Drivers  --->
        [*] Staging drivers  --->
            <*>     Industrial I/O support --->
                --- Industrial I/O support
                -*-   Enable ring buffer support within IIO
                -*-     Industrial I/O lock free software ring
                -*-   Enable triggered sampling support

                      *** Analog to digital converters ***
                [--snip--]

                <*>   Analog Devices AD7606 ADC driver
                <*>     parallel interface support
                <*>     spi interface support

                [--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:trigger0

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

root:/sys/bus/iio/devices/iio:device0> ls -l
drwxr-xr-x    5 root     root             0 Jan  1 00:00 buffer
-r--r--r--    1 root     root          4096 Jan  1 00:00 in_voltage0_raw
-r--r--r--    1 root     root          4096 Jan  1 00:00 in_voltage1_raw
-r--r--r--    1 root     root          4096 Jan  1 00:00 in_voltage2_raw
-r--r--r--    1 root     root          4096 Jan  1 00:00 in_voltage3_raw
-r--r--r--    1 root     root          4096 Jan  1 00:00 in_voltage4_raw
-r--r--r--    1 root     root          4096 Jan  1 00:00 in_voltage5_raw
-r--r--r--    1 root     root          4096 Jan  1 00:00 in_voltage6_raw
-r--r--r--    1 root     root          4096 Jan  1 00:00 in_voltage7_raw
-r--r--r--    1 root     root          4096 Jan  1 00:00
in_voltage_scale
-r--r--r--    1 root     root          4096 Jan  1 00:00 name
-rw-r--r--    1 root     root          4096 Jan  1 00:00
oversampling_ratio
-r--r--r--    1 root     root          4096 Jan  1 00:00
oversampling_ratio_available
-rw-r--r--    1 root     root          4096 Jan  1 00:00 range
-r--r--r--    1 root     root          4096 Jan  1 00:00 range_available
lrwxrwxrwx    1 root     root             0 Jan  1 00:00 subsystem ->
../../../../bus/iio
drwxr-xr-x    2 root     root             0 Jan  1 00:00 trigger
-rw-r--r--    1 root     root          4096 Jan  1 00:00 uevent
root:/sys/bus/iio/devices/iio:device0>
```

Show device name

This specifies any shell prompt running on the target

```
root:/sys/bus/iio/devices/iio:device0> cat name
ad7606
```

Show available oversampling ratios

This specifies any shell prompt running on the target

```
root:/sys/bus/iio/devices/iio:device0> cat oversampling_ratio_available
0 2 4 8 16 32 64
```

Show available input ranges

This specifies any shell prompt running on the target

```
root:/sys/bus/iio/devices/iio:device0> cat range_available
5000 10000
```

Set input range to 10Volt

This specifies any shell prompt running on the target

```
root:/sys/bus/iio/devices/iio:device0> echo 10000 > range
root:/sys/bus/iio/devices/iio:device0> cat range
10000
```

Show scale

Description:
scale to be applied to in0_raw in order to obtain the measured voltage in millivolts.

This specifies any shell prompt running on the target

```
root:/sys/bus/iio/devices/iio:device0> cat in_voltage_scale
0.152
```

Show channel 2 measurement

Description:
Raw unscaled voltage measurement on channel 2

This specifies any shell prompt running on the target

```
root:/sys/bus/iio/devices/iio:device0> cat in_voltage2_raw
5789
```

U = *in2_raw* * *in_scale* = 5789 * 0.152 = ***879,928 mV***

Trigger management

If deviceX supports triggered sampling, it's a so called trigger consumer and there will be an additional folder /sys/bus/iio/device/iio:deviceX/trigger. In this folder there is a file called current_trigger, allowing controlling and viewing the current trigger source connected to deviceX. Available trigger sources can be identified by reading the name file /sys/bus/iio/devices/triggerY/name. The same trigger source can connect to multiple devices, so a single trigger may initialize data capture or reading from a number of sensors, converters, etc.

Trigger Consumers:
Currently triggers are only used for the filling of software ring buffers and as such any device supporting INDIO_RING_TRIGGERED has the consumer interface automatically created.

Description: Read name of triggerY

This specifies any shell prompt running on the target

```
root:/sys/bus/iio/devices/triggerY/> cat name
irqtrig56
```

Description: Make irqtrig56 (trigger using system IRQ56, likely a GPIO IRQ), to current trigger of deviceX

This specifies any shell prompt running on the target

```
root:/sys/bus/iio/devices/iio:deviceX/trigger> echo irqtrig56 >
current_trigger
```

Description: Read current trigger source of deviceX

This specifies any shell prompt running on the target

```
root:/sys/bus/iio/devices/iio:deviceX/trigger> cat current_trigger
irqtrig56
```

02 Mar 2011 14:16 · Michael Hennerich

Available standalone trigger drivers

name	description
iio-trig-gpio	Provides support for using GPIO pins as IIO triggers.
iio-trig-rtc	Provides support for using periodic capable real time clocks as IIO triggers.
iio-trig-sysfs	Provides support for using SYSFS entry as IIO triggers.
iio-trig-bfin-timer	Provides support for using a Blackfin timer as IIO triggers.

02 Mar 2011 14:16 · Michael Hennerich

Buffer management

This specifies any shell prompt running on the target

```
root:/sys/bus/iio/devices/iio:device0/buffer> ls
enable                    subsystem
length                    uevent
root:/sys/bus/iio/devices/iio:device0/buffer>
```

The Industrial I/O subsystem provides support for various ring buffer based data acquisition methods. Apart from device specific hardware buffer support, the user can chose between two different software ring buffer implementations. One is the IIO lock free software ring, and the other is based on Linux kfifo. Devices with buffer support feature an additional sub-folder in the /sys/bus/iio/devices/deviceX/ folder hierarchy. Called deviceX:bufferY, where Y defaults to 0, for devices with a single buffer.

Every buffer implementation features a set of files:

length
Get/set the number of sample sets that may be held by the buffer.

enable
Enables/disables the buffer. This file should be written last, after length and selection of scan elements.

scan_elements
The scan_elements directory contains interfaces for elements that will be captured for a single triggered sample set in the buffer.

02 Mar 2011 14:16 · Michael Hennerich

This specifies any shell prompt running on the target

```
root:/sys/bus/iio/devices/iio:device0/scan_elements> ls
in_voltage0_en            in_voltage2_index         in_voltage5_en
in_voltage7_index
in_voltage0_index         in_voltage3_en            in_voltage5_index
in_voltage_type
in_voltage1_en            in_voltage3_index         in_voltage6_en
timestamp_en
```

```
in_voltage1_index        in_voltage4_en           in_voltage6_index
timestamp_index
in_voltage2_en           in_voltage4_index        in_voltage7_en
timestamp_type
root:/sys/bus/iio/devices/iio:device0/scan_elements>
```

in_voltageX_en / in_voltageX-voltageY_en / timestamp_en:
Scan element control for triggered data capture. Writing 1 will enable the scan element, writing 0 will disable it

in_voltageX_type / in_voltageX-voltageY_type / timestamp_type:
Description of the scan element data storage within the buffer and therefore in the form in which it is read from user-space. Form is [s|u]bits/storage-bits. s or u specifies if signed (2's complement) or unsigned. bits is the number of bits of data and storage-bits is the space (after padding) that it occupies in the buffer. Note that some devices will have additional information in the unused bits so to get a clean value, the bits value must be used to mask the buffer output value appropriately. The storage-bits value also specifies the data alignment. So u12/16 will be a unsigned 12 bit integer stored in a 16 bit location aligned to a 16 bit boundary. For other storage combinations this attribute will be extended appropriately.

in_voltageX_index / in_voltageX-voltageY_index / timestamp_index:
A single positive integer specifying the position of this scan element in the buffer. Note these are not dependent on what is enabled and may not be contiguous. Thus for user-space to establish the full layout these must be used in conjunction with all _en attributes to establish which channels are present, and the relevant _type attributes to establish the data storage format.

02 Mar 2011 14:16 · Michael Hennerich

More Information

- IIO mailing list: linux-iio@vger.kernel.org
- IIO Documentation
- IIO Utils Main Page
- IIO test and visualization application
- libiio - IIO system library
- Pointers and good books
- Video from Fosdem of how IIO is used in SDR applications

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

www.analog.com