Core elements

The Industrial I/O core offers both a unified framework for writing drivers for many different types of embedded sensors and a standard interface to user space applications manipulating sensors. The implementation can be found under :file:`drivers/iio/industrialio-*`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 5); *backlink*

Unknown interpreted text role "file".

Industrial I/O Devices

- struct iio_dev - industrial I/O device
- iio_device_alloc() - allocate an :c:type:`iio_dev` from a driver

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 14); *backlink*

Unknown interpreted text role "c:type".

- iio_device_free() - free an :c:type:`iio_dev` from a driver

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 15); *backlink*

Unknown interpreted text role "c:type".

- iio_device_register() - register a device with the IIO subsystem
- iio_device_unregister() - unregister a device from the IIO subsystem

An IIO device usually corresponds to a single hardware sensor and it provides all the information needed by a driver handling a device. Let's first have a look at the functionality embedded in an IIO device then we will show how a device driver makes use of an IIO device.

There are two ways for a user space application to interact with an IIO driver.

1. :file:`/sys/bus/iio/iio:device{X}/`, this represents a hardware sensor and groups together the data channels of the same chip.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 27); *backlink*

Unknown interpreted text role "file".

2. :file:`/dev/iio:device{X}`, character device node interface used for buffered data transfer and for events information retrieval.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 29); *backlink*

Unknown interpreted text role "file".

A typical IIO driver will register itself as an :doc:`I2C <../i2c>` or :doc:`SPI <../spi>` driver and will create two routines, probe and remove.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 32); *backlink*

Unknown interpreted text role "doc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 32); *backlink*

Unknown interpreted text role "doc".

At probe:

1. Call iio_device_alloc(), which allocates memory for an IIO device.
2. Initialize IIO device fields with driver specific information (e.g. device name, device channels).
3. Call iio_device_register(), this registers the device with the IIO core. After this call the device is ready to accept requests from user space applications.

At remove, we free the resources allocated in probe in reverse order:

1. iio_device_unregister(), unregister the device from the IIO core.
2. iio_device_free(), free the memory allocated for the IIO device.

IIO device sysfs interface

Attributes are sysfs files used to expose chip info and also allowing applications to set various configuration parameters. For device with index X, attributes can be found under /sys/bus/iio/iio:deviceX/ directory. Common attributes are:

- :file:`name`, description of the physical chip.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 57); *backlink*

  Unknown interpreted text role "file".

- :file:`dev`, shows the major:minor pair associated with :file:`/dev/iio:deviceX` node.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 58); *backlink*

  Unknown interpreted text role "file".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 58); *backlink*

  Unknown interpreted text role "file".

- :file:`sampling_frequency_available`, available discrete set of sampling frequency values for device.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 60); *backlink*

  Unknown interpreted text role "file".

- Available standard attributes for IIO devices are described in the :file:`Documentation/ABI/testing/sysfs-bus-iio` file in the Linux kernel sources.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 62); *backlink*

  Unknown interpreted text role "file".

IIO device channels

struct iio_chan_spec - specification of a single channel

An IIO device channel is a representation of a data channel. An IIO device can have one or multiple channels. For example:

- a thermometer sensor has one channel representing the temperature measurement.

- a light sensor with two channels indicating the measurements in the visible and infrared spectrum.
- an accelerometer can have up to 3 channels representing acceleration on X, Y and Z axes.

An IIO channel is described by the struct iio_chan_spec. A thermometer driver for the temperature sensor in the example above would have to describe its channel as follows:

```
static const struct iio_chan_spec temp_channel[] = {
     {
          .type = IIO_TEMP,
          .info_mask_separate = BIT(IIO_CHAN_INFO_PROCESSED),
     },
};
```

Channel sysfs attributes exposed to userspace are specified in the form of bitmasks. Depending on their shared info, attributes can be set in one of the following masks:

- **info_mask_separate**, attributes will be specific to this channel
- **info_mask_shared_by_type**, attributes are shared by all channels of the same type
- **info_mask_shared_by_dir**, attributes are shared by all channels of the same direction
- **info_mask_shared_by_all**, attributes are shared by all channels

When there are multiple data channels per channel type we have two ways to distinguish between them:

- set **.modified** field of :c:type:`iio_chan_spec` to 1. Modifiers are specified using **.channel2** field of the same :c:type:`iio_chan_spec` structure and are used to indicate a physically unique characteristic of the channel such as its direction or spectral response. For example, a light sensor can have two channels, one for infrared light and one for both infrared and visible light.

  > **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, **line 106**); *backlink*
  >
  > Unknown interpreted text role "c:type".

  > **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, **line 106**); *backlink*
  >
  > Unknown interpreted text role "c:type".

- set **.indexed** field of :c:type:`iio_chan_spec` to 1. In this case the channel is simply another instance with an index specified by the **.channel** field.

  > **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, **line 112**); *backlink*
  >
  > Unknown interpreted text role "c:type".

Here is how we can make use of the channel's modifiers:

```
static const struct iio_chan_spec light_channels[] = {
        {
                .type = IIO_INTENSITY,
                .modified = 1,
                .channel2 = IIO_MOD_LIGHT_IR,
                .info_mask_separate = BIT(IIO_CHAN_INFO_RAW),
                .info_mask_shared = BIT(IIO_CHAN_INFO_SAMP_FREQ),
        },
        {
                .type = IIO_INTENSITY,
                .modified = 1,
                .channel2 = IIO_MOD_LIGHT_BOTH,
                .info_mask_separate = BIT(IIO_CHAN_INFO_RAW),
                .info_mask_shared = BIT(IIO_CHAN_INFO_SAMP_FREQ),
        },
        {
                .type = IIO_LIGHT,
                .info_mask_separate = BIT(IIO_CHAN_INFO_PROCESSED),
                .info_mask_shared = BIT(IIO_CHAN_INFO_SAMP_FREQ),
        },
   }
```

This channel's definition will generate two separate sysfs files for raw data retrieval:

- :file:`/sys/bus/iio/iio:device{X}/in_intensity_ir_raw`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 143); *backlink*

  Unknown interpreted text role "file".

- :file:`/sys/bus/iio/iio:device{X}/in_intensity_both_raw`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 144); *backlink*

  Unknown interpreted text role "file".

one file for processed data:

- :file:`/sys/bus/iio/iio:device{X}/in_illuminance_input`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 148); *backlink*

  Unknown interpreted text role "file".

and one shared sysfs file for sampling frequency:

- :file:`/sys/bus/iio/iio:device{X}/sampling_frequency`.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 152); *backlink*

  Unknown interpreted text role "file".

Here is how we can make use of the channel's indexing:

```
static const struct iio_chan_spec light_channels[] = {
        {
                .type = IIO_VOLTAGE,
                .indexed = 1,
                .channel = 0,
                .info_mask_separate = BIT(IIO_CHAN_INFO_RAW),
        },
        {
                .type = IIO_VOLTAGE,
                .indexed = 1,
                .channel = 1,
                .info_mask_separate = BIT(IIO_CHAN_INFO_RAW),
        },
}
```

This will generate two separate attributes files for raw data retrieval:

- :file:`/sys/bus/iio/devices/iio:device{X}/in_voltage0_raw`, representing voltage measurement for channel 0.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 173); *backlink*

  Unknown interpreted text role "file".

- :file:`/sys/bus/iio/devices/iio:device{X}/in_voltage1_raw`, representing voltage measurement for channel 1.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 175); *backlink*

  Unknown interpreted text role "file".

More details

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 180)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/linux/iio/iio.h
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\(linux-master)(Documentation)(driver-api)(iio)core.rst, line 181)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/iio/industrialio-core.c
   :export:
```