Triggers

- struct iio_trigger â€” industrial I/O trigger device
- :c:func:`devm_iio_trigger_alloc` â€” Resource-managed iio_trigger_alloc

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\[linux-master][Documentation][driver-api][iio]triggers.rst, line 6); *backlink*

  Unknown interpreted text role "c:func".

- :c:func:`devm_iio_trigger_register` â€” Resource-managed iio_trigger_register iio_trigger_unregister

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\[linux-master][Documentation][driver-api][iio]triggers.rst, line 7); *backlink*

  Unknown interpreted text role "c:func".

- :c:func:`iio_trigger_validate_own_device` â€” Check if a trigger and IIO device belong to the same device

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\[linux-master][Documentation][driver-api][iio]triggers.rst, line 9); *backlink*

  Unknown interpreted text role "c:func".

In many situations it is useful for a driver to be able to capture data based on some external event (trigger) as opposed to periodically polling for data. An IIO trigger can be provided by a device driver that also has an IIO device based on hardware generated events (e.g. data ready or threshold exceeded) or provided by a separate driver from an independent interrupt source (e.g. GPIO line connected to some external system, timer interrupt or user space writing a specific file in sysfs). A trigger may initiate data capture for a number of sensors and also it may be completely unrelated to the sensor itself.

IIO trigger sysfs interface

There are two locations in sysfs related to triggers:

- :file:`/sys/bus/iio/devices/trigger{Y}/*`, this file is created once an IIO trigger is registered with the IIO core and corresponds to trigger with index Y. Because triggers can be very different depending on type there are few standard attributes that we can describe here:

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\[linux-master][Documentation][driver-api][iio]triggers.rst, line 26); *backlink*

  Unknown interpreted text role "file".

  - :file:`name`, trigger name that can be later used for association with a device.

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

    Unknown interpreted text role "file".

  - :file:`sampling_frequency`, some timer based triggers use this attribute to specify the frequency for trigger calls.

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\[linux-master][Documentation][driver-api][iio]triggers.rst, line 34); *backlink*

    Unknown interpreted text role "file".

- :file:`/sys/bus/iio/devices/iio:device{X}/trigger/*`, this directory is created once the device supports a triggered buffer. We can

associate a trigger with our device by writing the trigger's name in the :file:`current_trigger` file.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\[linux-master][Documentation][driver-api][iio]triggers.rst, line 37); *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]triggers.rst, line 37); *backlink*

Unknown interpreted text role "file".

IIO trigger setup

Let's see a simple example of how to setup a trigger to be used by a driver:

```
struct iio_trigger_ops trigger_ops = {
    .set_trigger_state = sample_trigger_state,
    .validate_device = sample_validate_device,
}

struct iio_trigger *trig;

/* first, allocate memory for our trigger */
trig = iio_trigger_alloc(dev, "trig-%s-%d", name, idx);

/* setup trigger operations field */
trig->ops = &trigger_ops;

/* now register the trigger with the IIO core */
iio_trigger_register(trig);
```

IIO trigger ops

- struct iio_trigger_ops â€” operations structure for an iio_trigger.

Notice that a trigger has a set of operations attached:

- :file:`set_trigger_state`, switch the trigger on/off on demand.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\[linux-master][Documentation][driver-api][iio]triggers.rst, line 70); *backlink*

Unknown interpreted text role "file".

- :file:`validate_device`, function to validate the device when the current trigger gets changed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\[linux-master][Documentation][driver-api][iio]triggers.rst, line 71); *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]triggers.rst, line 76)

Unknown directive type "kernel-doc".

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\iio\[linux-master][Documentation][driver-api]

[iio]triggers.rst, line 77)

Unknown directive type "kernel-doc".

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