V4L2 events

The V4L2 events provide a generic way to pass events to user space. The driver must use :c:type:`v4l2_fh` to be able to support V4L2 events.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 6**); ***backlink***

Unknown interpreted text role "c:type".

Events are subscribed per-filehandle. An event specification consists of a `type` and is optionally associated with an object identified through the `id` field. If unused, then the `id` is 0. So an event is uniquely identified by the `(type, id)` tuple.

The :c:type:`v4l2_fh` struct has a list of subscribed events on its `subscribed` field.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 14**); ***backlink***

Unknown interpreted text role "c:type".

When the user subscribes to an event, a :c:type:`v4l2_subscribed_event` struct is added to :c:type:`v4l2_fh`.`subscribed`, one for every subscribed event.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 17**); ***backlink***

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 17**); ***backlink***

Unknown interpreted text role "c:type".

Each :c:type:`v4l2_subscribed_event` struct ends with a :c:type:`v4l2_kevent` ringbuffer, with the size given by the caller of :c:func:`v4l2_event_subscribe`. This ringbuffer is used to store any events raised by the driver.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 21**); ***backlink***

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 21**); ***backlink***

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 21**); ***backlink***

Unknown interpreted text role "c:func".

So every `(type, ID)` event tuple will have its own :c:type:`v4l2_kevent` ringbuffer. This guarantees that if a driver is generating lots of events of one type in a short time, then that will not overwrite events of another type.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api]` [media]v4l2-event.rst, line 26); *backlink*

Unknown interpreted text role "c:type".

But if you get more events of one type than the size of the :c:type:`v4l2_kevent` ringbuffer, then the oldest event will be dropped and the new one added.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst, line 31); *backlink*

Unknown interpreted text role "c:type".

The :c:type:`v4l2_kevent` struct links into the `available` list of the :c:type:`v4l2_fh` struct so :ref:`VIDIOC_DQEVENT` will know which event to dequeue first.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst, line 35); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst, line 35); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst, line 35); *backlink*

Unknown interpreted text role "ref".

Finally, if the event subscription is associated with a particular object such as a V4L2 control, then that object needs to know about that as well so that an event can be raised by that object. So the `node` field can be used to link the :c:type:`v4l2_subscribed_event` struct into a list of such objects.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst, line 39); *backlink*

Unknown interpreted text role "c:type".

So to summarize:

- struct v4l2_fh has two lists: one of the `subscribed` events, and one of the `available` events.
- struct v4l2_subscribed_event has a ringbuffer of raised (pending) events of that particular type.
- If struct v4l2_subscribed_event is associated with a specific object, then that object will have an internal list of struct v4l2_subscribed_event so it knows who subscribed an event to that object.

Furthermore, the internal struct v4l2_subscribed_event has `merge()` and `replace()` callbacks which drivers can set. These callbacks are called when a new event is raised and there is no more room.

The `replace()` callback allows you to replace the payload of the old event with that of the new event, merging any relevant data from the old payload into the new payload that replaces it. It is called when this event type has a ringbuffer with size is one, i.e. only one event can be stored in the ringbuffer.

The `merge()` callback allows you to merge the oldest event payload into that of the second-oldest event payload. It is called when the ringbuffer has size is greater than one.

This way no status information is lost, just the intermediate steps leading up to that state.

A good example of these `replace/merge` callbacks is in v4l2-event.c: `ctrls_replace()` and `ctrls_merge()` callbacks for the control event.

Note

these callbacks can be called from interrupt context, so they must be fast.

In order to queue events to video device, drivers should call:

> :c:func:`v4l2_event_queue <v4l2_event_queue>` (:c:type:`vdev <video_device>`, :c:type:`ev <v4l2_event>`)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 84**); ***backlink***

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 84**); ***backlink***

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 84**); ***backlink***

Unknown interpreted text role "c:type".

The driver's only responsibility is to fill in the type and the data fields. The other fields will be filled in by V4L2.

Event subscription

Subscribing to an event is via:

> :c:func:`v4l2_event_subscribe <v4l2_event_subscribe>` (:c:type:`fh <v4l2_fh>`, :c:type:`sub <v4l2_event_subscription>` , elems, :c:type:`ops <v4l2_subscribed_event_ops>`)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 95**); ***backlink***

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 95**); ***backlink***

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 95**); ***backlink***

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 95**); ***backlink***

Unknown interpreted text role "c:type".

This function is used to implement :c:type:`video_device`-> :c:type:`ioctl_ops <v4l2_ioctl_ops>`-> `vidioc_subscribe_event`, but the driver must check first if the driver is able to produce events with specified event id, and then should call :c:func:`v4l2_event_subscribe` to subscribe the event.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, **line 100**); ***backlink***

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 100); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 100); *backlink*

Unknown interpreted text role "c:func".

The elems argument is the size of the event queue for this event. If it is 0, then the framework will fill in a default value (this depends on the event type).

The ops argument allows the driver to specify a number of callbacks:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 112)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{1.5cm}|p{16.0cm}|
```

Callback	Description
add	called when a new listener gets added (subscribing to the same event twice will only cause this callback to get called once)
del	called when a listener stops listening
replace	replace event 'old' with event 'new'.
merge	merge event 'old' into event 'new'.

All 4 callbacks are optional, if you don't want to specify any callbacks the ops argument itself maybe `NULL`.

Unsubscribing an event

Unsubscribing to an event is via:

:c:func:`v4l2_event_unsubscribe <v4l2_event_unsubscribe>` (:c:type:`fh <v4l2_fh>`, :c:type:`sub <v4l2_event_subscription>`)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 132); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 132); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 132); *backlink*

Unknown interpreted text role "c:type".

This function is used to implement :c:type:`video_device`-> :c:type:`ioctl_ops <v4l2_ioctl_ops>`-> `vidioc_unsubscribe_event`. A driver may call :c:func:`v4l2_event_unsubscribe` directly unless it wants to be involved in unsubscription process.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 135); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 135); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 135); *backlink*

Unknown interpreted text role "c:func".

The special type `V4L2_EVENT_ALL` may be used to unsubscribe all events. The drivers may want to handle this in a special way.

Check if there's a pending event

Checking if there's a pending event is via:

:c:func:`v4l2_event_pending <v4l2_event_pending>` (:c:type:`fh <v4l2_fh>`)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 148); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 148); *backlink*

Unknown interpreted text role "c:type".

This function returns the number of pending events. Useful when implementing poll.

How events work

Events are delivered to user space through the poll system call. The driver can use :c:type:`v4l2_fh`->wait (a wait_queue_head_t) as the argument for `poll_wait()`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 158); *backlink*

Unknown interpreted text role "c:type".

There are standard and private events. New standard events must use the smallest available event type. The drivers must allocate their events from their own class starting from class base. Class base is `V4L2_EVENT_PRIVATE_START` + n * 1000 where n is the lowest available number. The first event type in the class is reserved for future use, so the first available event type is 'class base + 1'.

An example on how the V4L2 events may be used can be found in the OMAP 3 ISP driver (`drivers/media/platform/ti/omap3isp`).

A subdev can directly send an event to the :c:type:`v4l2_device` notify function with `V4L2_DEVICE_NOTIFY_EVENT`. This allows the bridge to map the subdev that sends the event to the video node(s) associated with the subdev that need to be informed about such an event.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 172); *backlink*

Unknown interpreted text role "c:type".

V4L2 event functions and data structures

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-event.rst`, line 180)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/media/v4l2-event.h
```