Zephyr® Project
Developer Summit

Zephyr® Project
Developer Summit

Introducing the Zephyr Input Subsystem

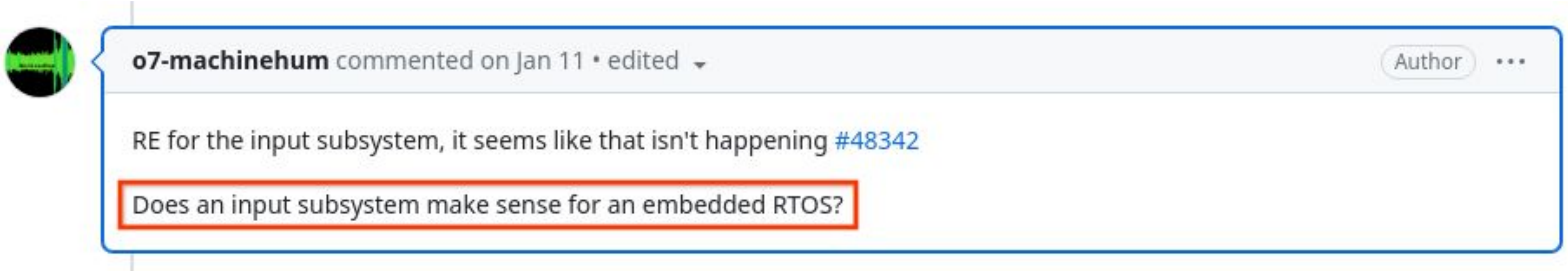
Fabio Baltieri, *Google*

fabiobaltieri@google.com
github.com/fabiobaltieri
fabiofw

#EMBEDDEDOSSUMMIT

Zephyr® Project
Developer Summit

Why?



o7-machinehum commented on Jan 11 • edited

RE for the input subsystem, it seems like that isn't happening #48342

Does an input subsystem make sense for an embedded RTOS?

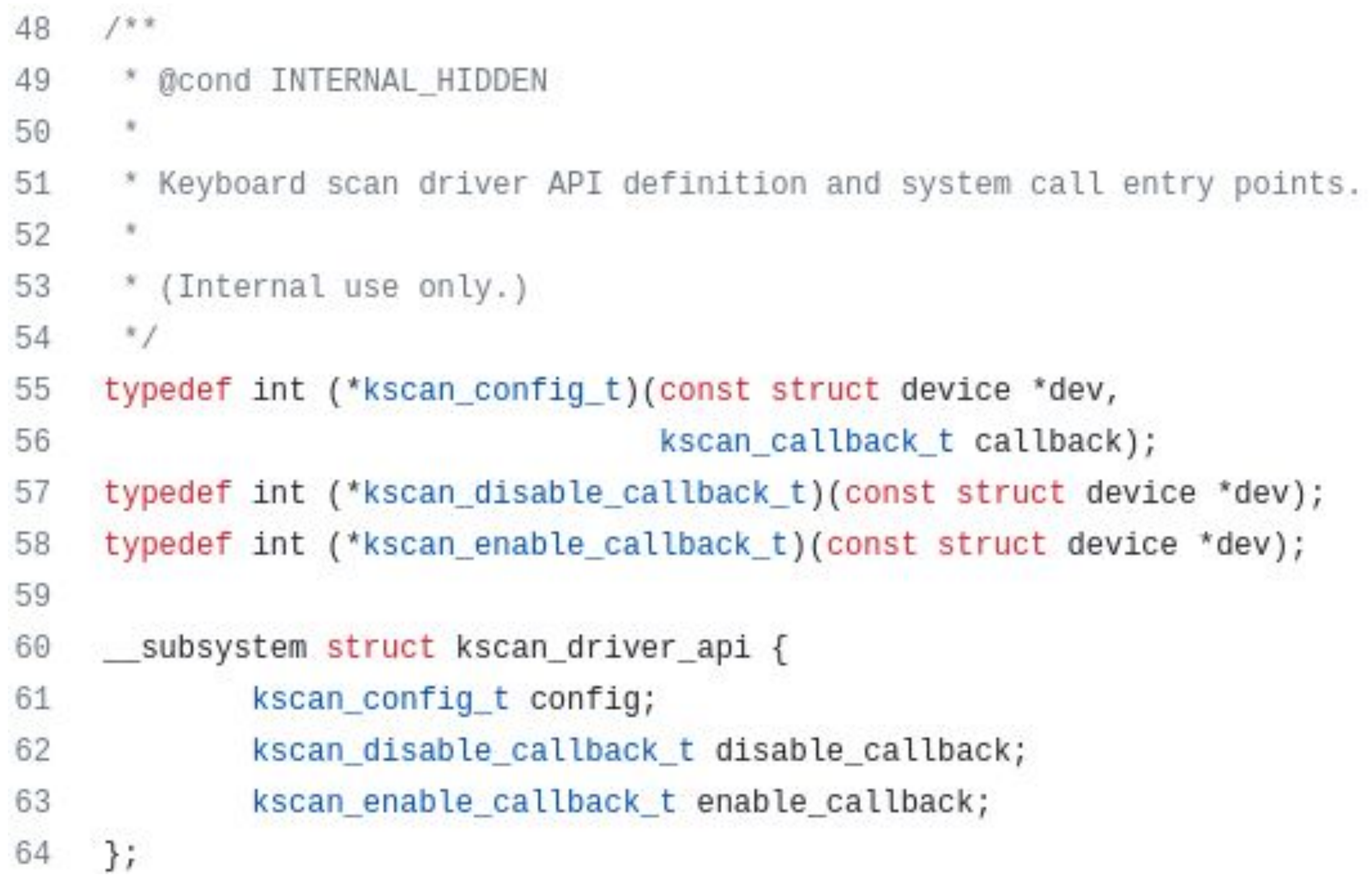
https://github.com/zephyrproject-rtos/zephyr/pull/52649#issuecomment-1379372241

Zephyr® Project
Developer Summit

Existing APIs

#EMBEDDEDOSSUMMIT

zephyr/include/zephyr/drivers/kscan.h

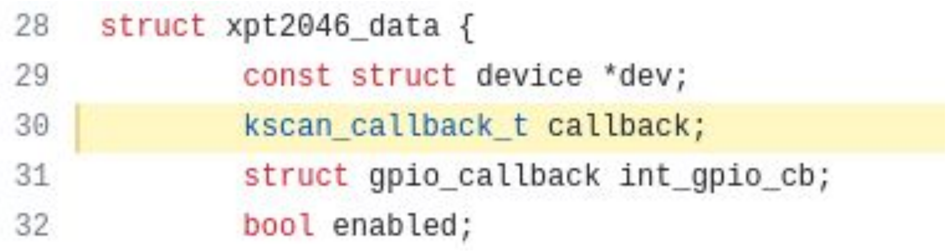


```c
/**
 * @cond INTERNAL_HIDDEN
 *
 * Keyboard scan driver API definition and system call entry points.
 *
 * (Internal use only.)
 */
typedef int (*kscan_config_t)(const struct device *dev,
                              kscan_callback_t callback);
typedef int (*kscan_disable_callback_t)(const struct device *dev);
typedef int (*kscan_enable_callback_t)(const struct device *dev);

__subsystem struct kscan_driver_api {
        kscan_config_t config;
        kscan_disable_callback_t disable_callback;
        kscan_enable_callback_t enable_callback;
};
```

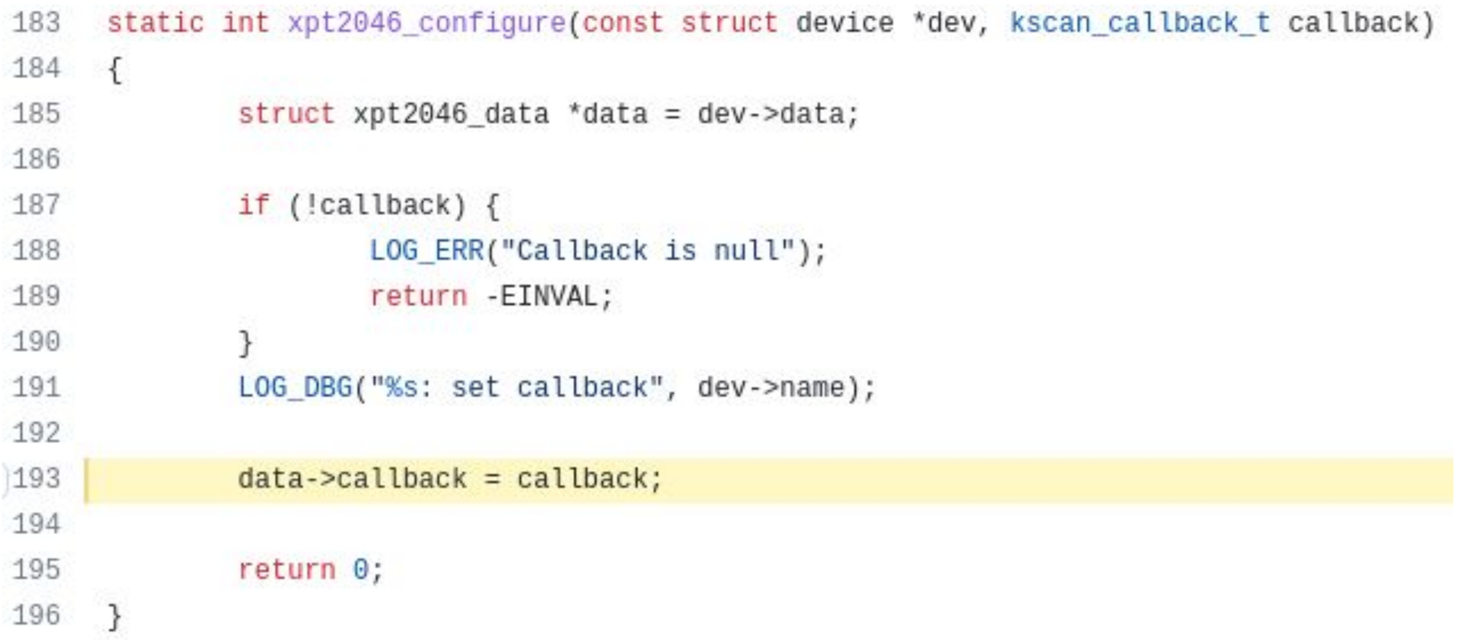
https://github.com/zephyrproject-rtos/zephyr/blob/main/include/zephyr/drivers/kscan.h#L48-L67

Zephyr® Project
Developer Summit

Single listener



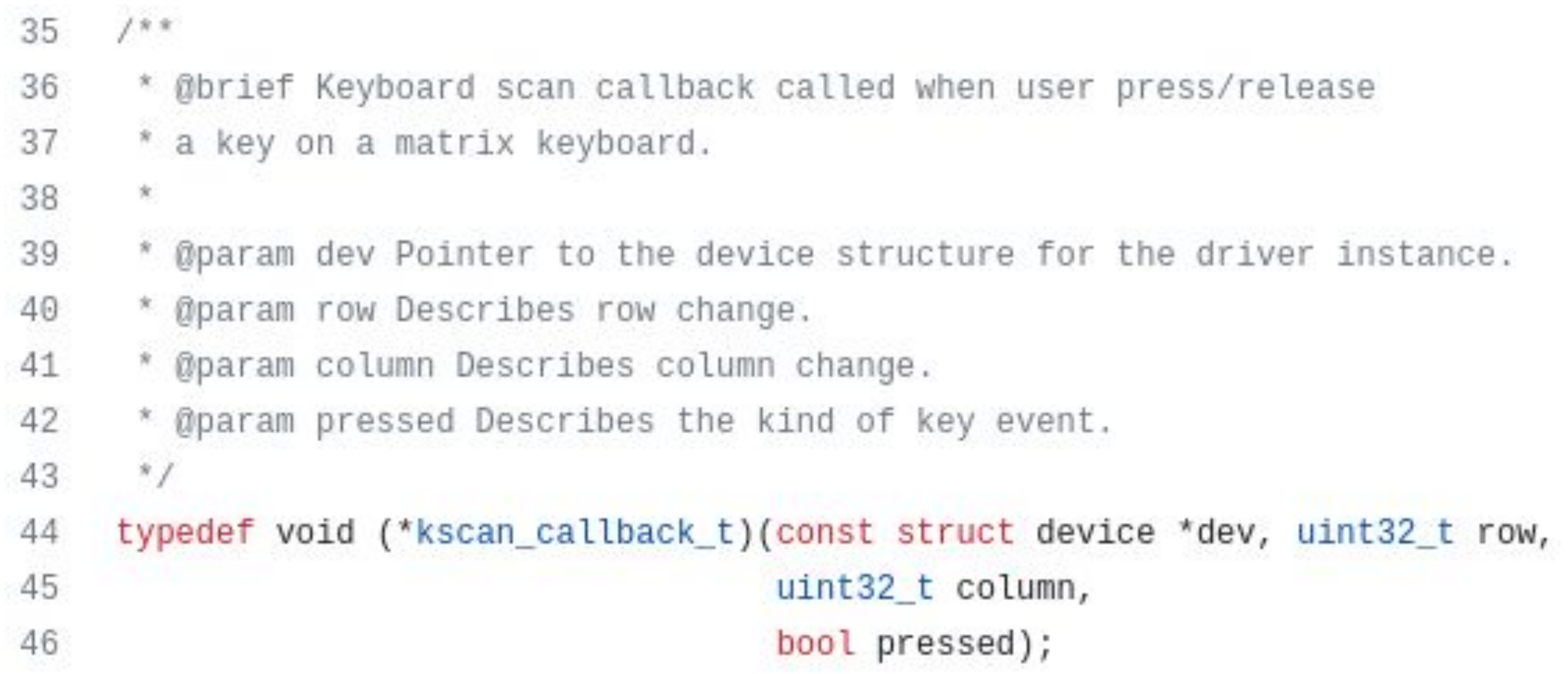
```
struct xpt2046_data {
	const struct device *dev;
	kscan_callback_t callback;
	struct gpio_callback int_gpio_cb;
	bool enabled;
```



```
static int xpt2046_configure(const struct device *dev, kscan_callback_t callback)
{
	struct xpt2046_data *data = dev->data;

	if (!callback) {
		LOG_ERR("Callback is null");
		return -EINVAL;
	}
	LOG_DBG("%s: set callback", dev->name);

	data->callback = callback;

	return 0;
}
```

https://github.com/zephyrproject-rtos/zephyr/blob/4babd545cc1b1ddef190aaedeabd1c3db3fbb8a2/drivers/kscan/kscan_xpt2046.c#L193

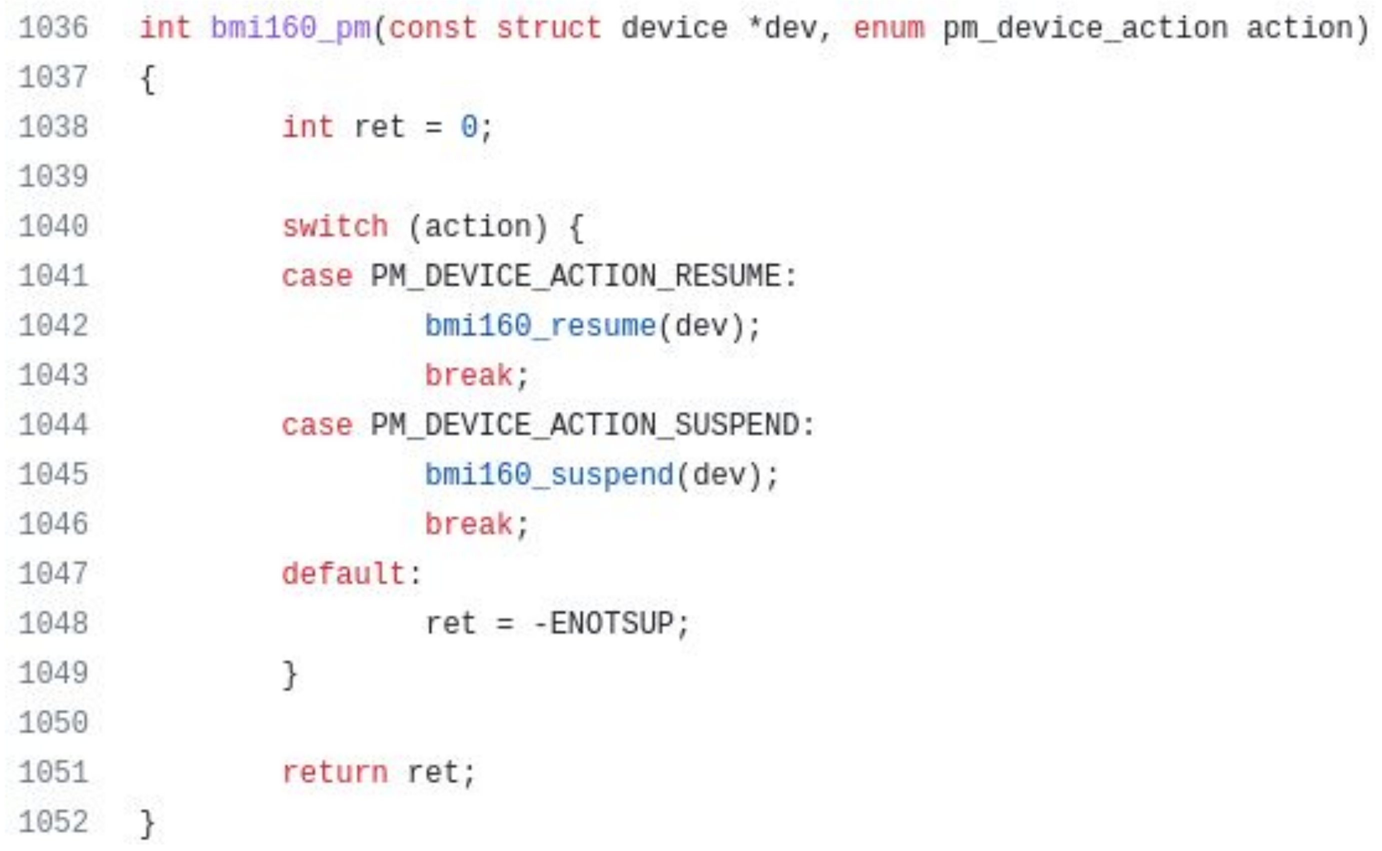
Limited use cases



```c
/**
 * @brief Keyboard scan callback called when user press/release
 * a key on a matrix keyboard.
 *
 * @param dev Pointer to the device structure for the driver instance.
 * @param row Describes row change.
 * @param column Describes column change.
 * @param pressed Describes the kind of key event.
 */
typedef void (*kscan_callback_t)(const struct device *dev, uint32_t row,
                                 uint32_t column,
                                 bool pressed);
```

https://github.com/zephyrproject-rtos/zephyr/blob/60a20471b561a0a3a74b377991fe1976c2ea83d7/include/zephyr/drivers/kscan.h#L44-L46

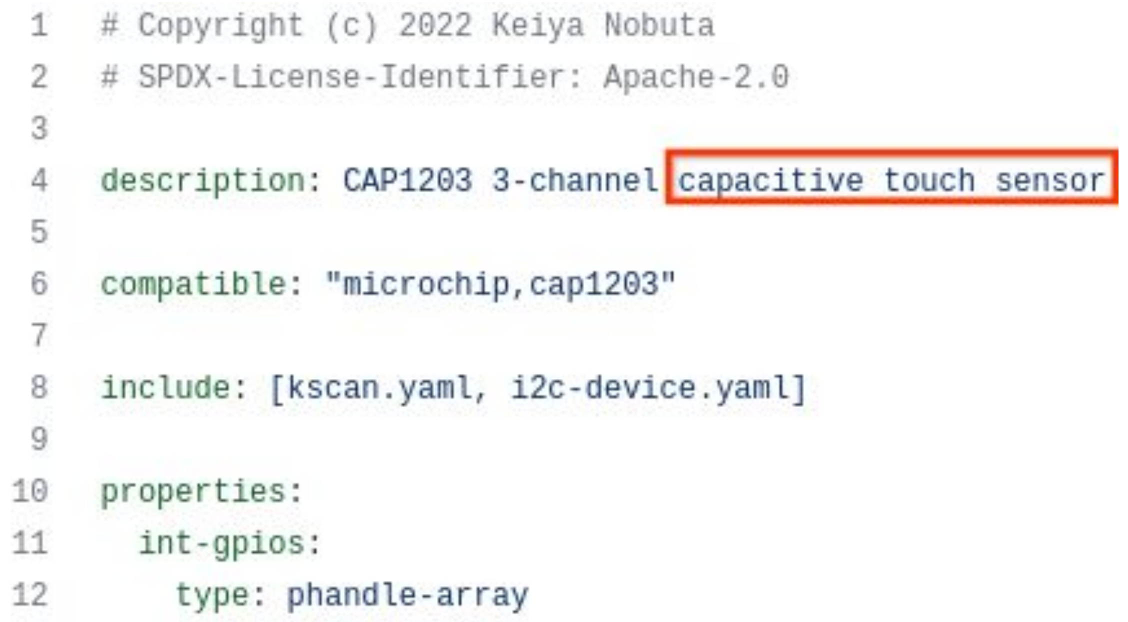
Redundant APIs



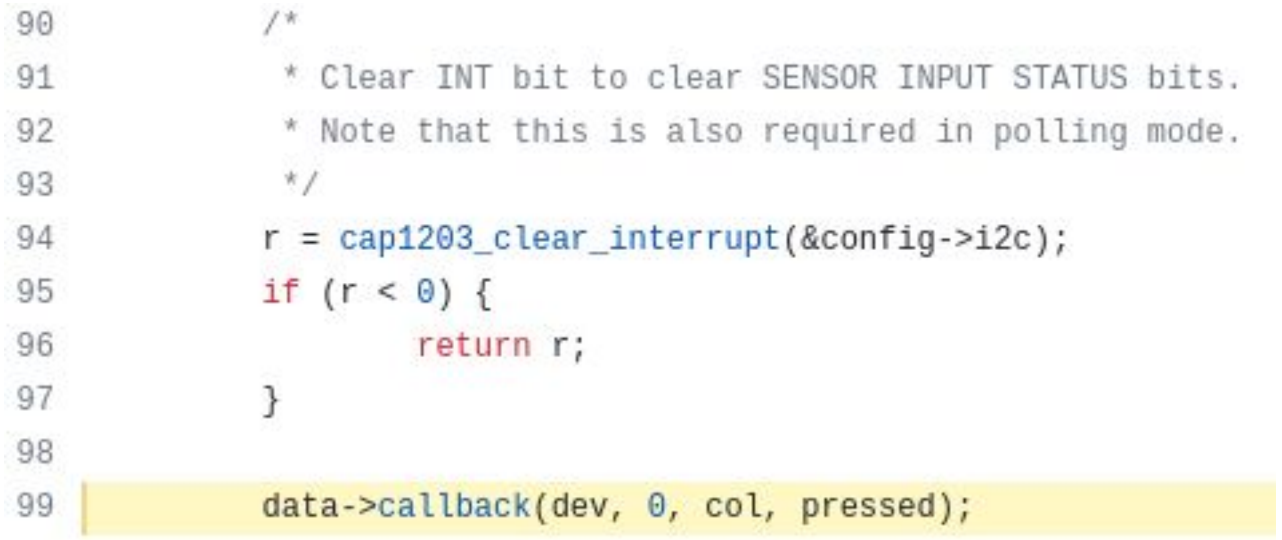
```
int bmi160_pm(const struct device *dev, enum pm_device_action action)
{
	int ret = 0;

	switch (action) {
	case PM_DEVICE_ACTION_RESUME:
		bmi160_resume(dev);
		break;
	case PM_DEVICE_ACTION_SUSPEND:
		bmi160_suspend(dev);
		break;
	default:
		ret = -ENOTSUP;
	}

	return ret;
}
```

https://github.com/zephyrproject-rtos/zephyr/blob/4babd545cc1b1ddef190aaedeabd1c3db3fbb8a2/drivers/sensor/bmi160/bmi160.c#L1036

Capsense



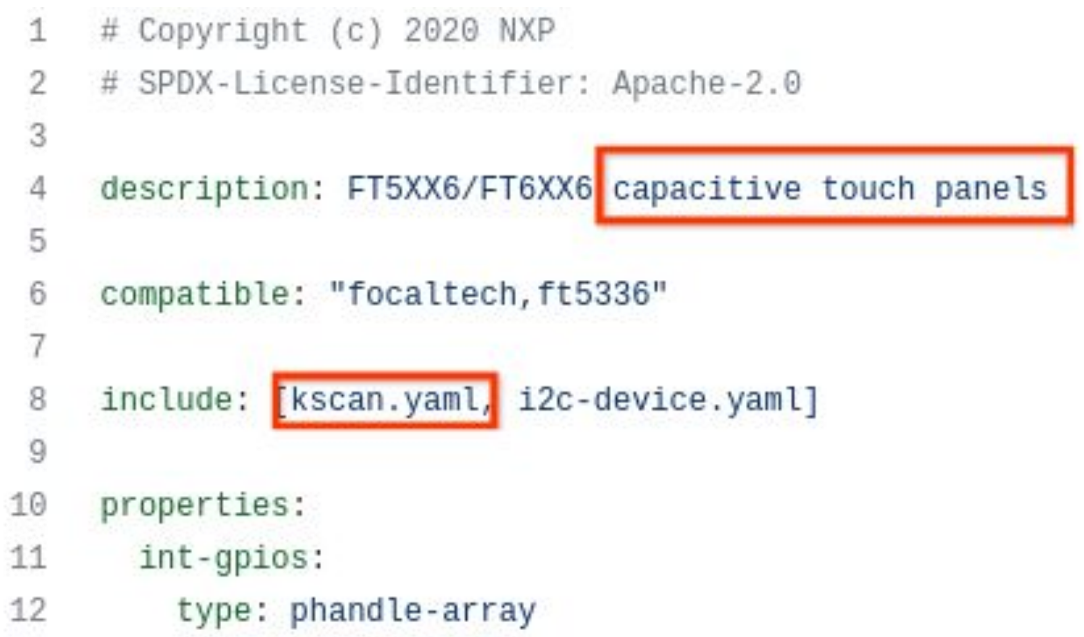
```
# Copyright (c) 2022 Keiya Nobuta
# SPDX-License-Identifier: Apache-2.0

description: CAP1203 3-channel capacitive touch sensor

compatible: "microchip,cap1203"

include: [kscan.yaml, i2c-device.yaml]

properties:
  int-gpios:
    type: phandle-array
```



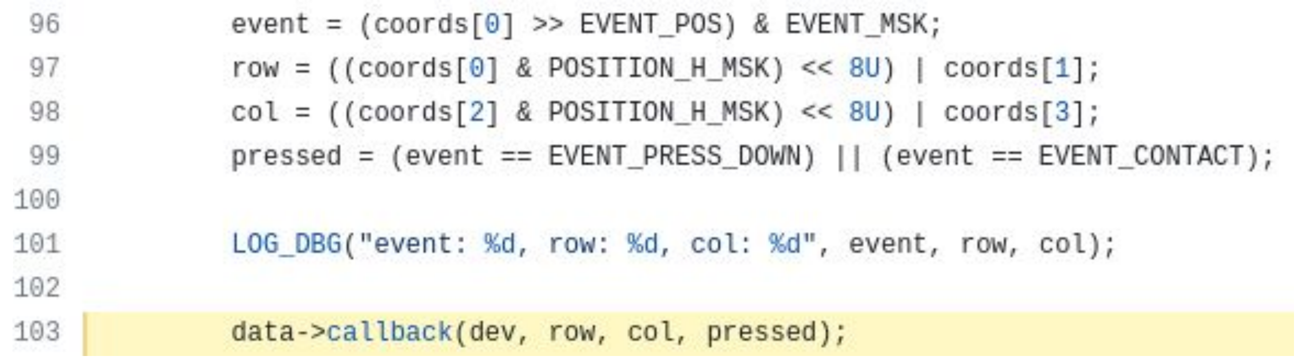
```
        /*
         * Clear INT bit to clear SENSOR INPUT STATUS bits.
         * Note that this is also required in polling mode.
         */
        r = cap1203_clear_interrupt(&config->i2c);
        if (r < 0) {
                return r;
        }

        data->callback(dev, 0, col, pressed);
```

https://github.com/zephyrproject-rtos/zephyr/blob/60a20471b561a0a3a74b377991fe1976c2ea83d7/drivers/kscan/kscan_cap1203.c#L99
https://github.com/zephyrproject-rtos/zephyr/blob/60a20471b561a0a3a74b377991fe1976c2ea83d7/dts/bindings/kscan/microchip%2Ccap1203.yaml

Row, column... it's a touchscreen!



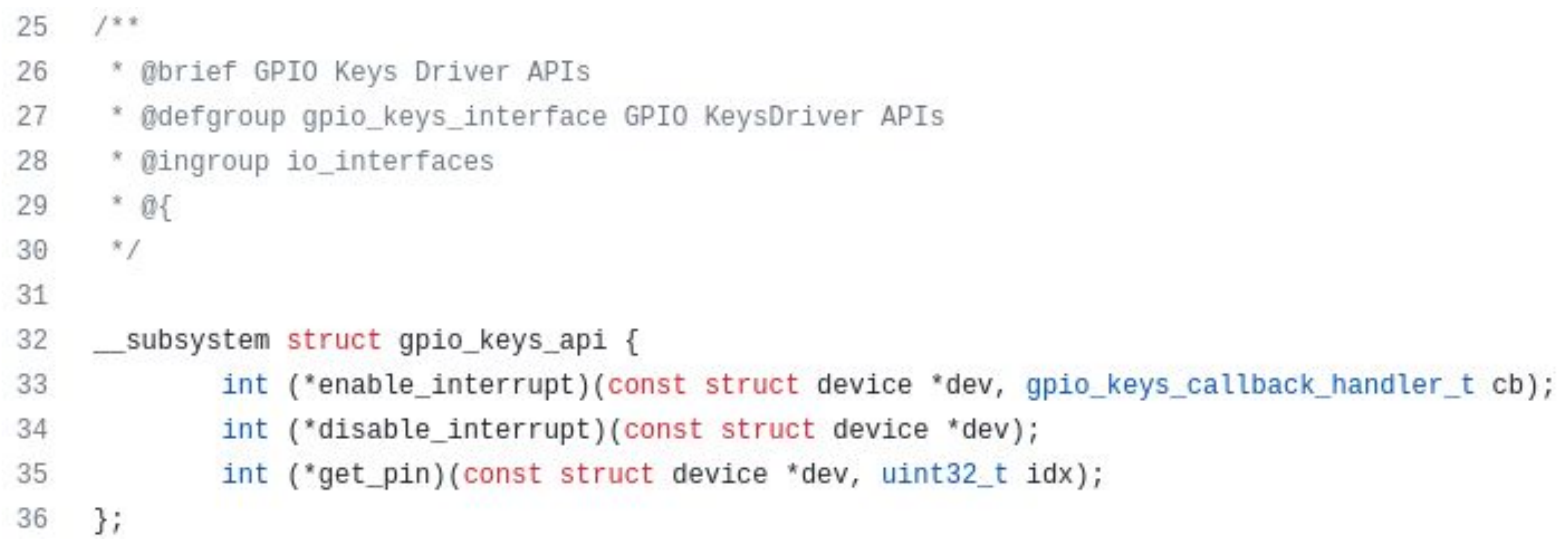
```yaml
# Copyright (c) 2020 NXP
# SPDX-License-Identifier: Apache-2.0

description: FT5XX6/FT6XX6 capacitive touch panels

compatible: "focaltech,ft5336"

include: [kscan.yaml, i2c-device.yaml]

properties:
  int-gpios:
    type: phandle-array
```



```c
        event = (coords[0] >> EVENT_POS) & EVENT_MSK;
        row = ((coords[0] & POSITION_H_MSK) << 8U) | coords[1];
        col = ((coords[2] & POSITION_H_MSK) << 8U) | coords[3];
        pressed = (event == EVENT_PRESS_DOWN) || (event == EVENT_CONTACT);

        LOG_DBG("event: %d, row: %d, col: %d", event, row, col);

        data->callback(dev, row, col, pressed);
```

https://github.com/zephyrproject-rtos/zephyr/blob/60a20471b561a0a3a74b377991fe1976c2ea83d7/dts/bindings/kscan/focaltech%2Cft5336.yaml
https://github.com/zephyrproject-rtos/zephyr/blob/60a20471b561a0a3a74b377991fe1976c2ea83d7/drivers/kscan/kscan_ft5336.c#L103

Don't even try

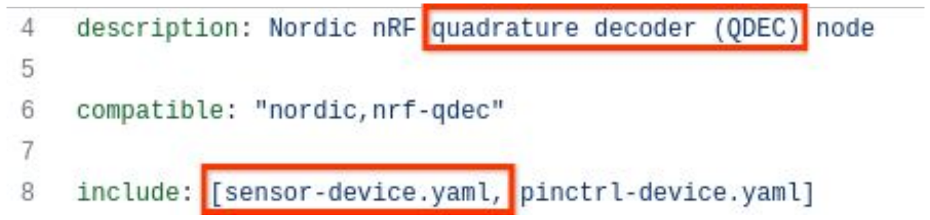


```
/**
 * @brief GPIO Keys Driver APIs
 * @defgroup gpio_keys_interface GPIO KeysDriver APIs
 * @ingroup io_interfaces
 * @{
 */

__subsystem struct gpio_keys_api {
        int (*enable_interrupt)(const struct device *dev, gpio_keys_callback_handler_t cb);
        int (*disable_interrupt)(const struct device *dev);
        int (*get_pin)(const struct device *dev, uint32_t idx);
};
```

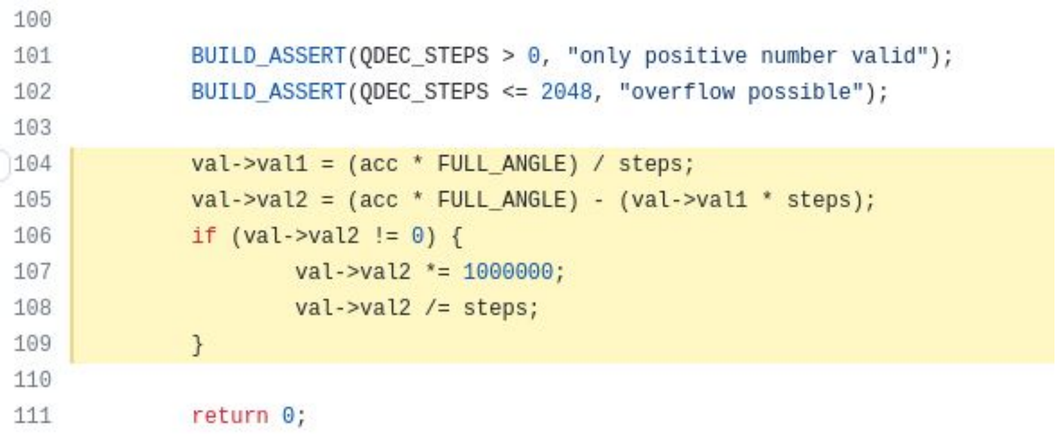
https://github.com/zephyrproject-rtos/zephyr/blob/60a20471b561a0a3a74b377991fe1976c2ea83d7/include/zephyr/drivers/gpio_keys.h#L32-L36

Zephyr® Project
Developer Summit

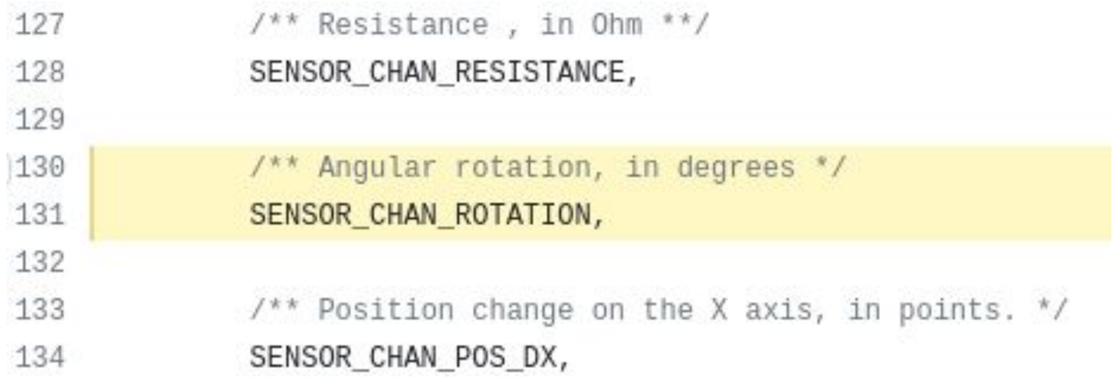
It's not an input device, it's a sensor



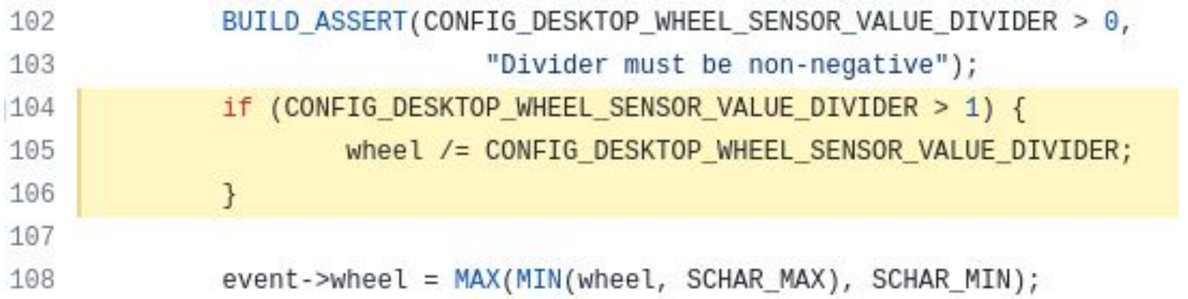
```
4   description: Nordic nRF quadrature decoder (QDEC) node
5
6   compatible: "nordic,nrf-qdec"
7
8   include: [sensor-device.yaml, pinctrl-device.yaml]
```



```
100
101         BUILD_ASSERT(QDEC_STEPS > 0, "only positive number valid");
102         BUILD_ASSERT(QDEC_STEPS <= 2048, "overflow possible");
103
104         val->val1 = (acc * FULL_ANGLE) / steps;
105         val->val2 = (acc * FULL_ANGLE) - (val->val1 * steps);
106         if (val->val2 != 0) {
107                 val->val2 *= 1000000;
108                 val->val2 /= steps;
109         }
110
111         return 0;
```



```
127         /** Resistance , in Ohm **/
128         SENSOR_CHAN_RESISTANCE,
129
130         /** Angular rotation, in degrees */
131         SENSOR_CHAN_ROTATION,
132
133         /** Position change on the X axis, in points. */
134         SENSOR_CHAN_POS_DX,
```



```
102         BUILD_ASSERT(CONFIG_DESKTOP_WHEEL_SENSOR_VALUE_DIVIDER > 0,
103                      "Divider must be non-negative");
104         if (CONFIG_DESKTOP_WHEEL_SENSOR_VALUE_DIVIDER > 1) {
105                 wheel /= CONFIG_DESKTOP_WHEEL_SENSOR_VALUE_DIVIDER;
106         }
107
108         event->wheel = MAX(MIN(wheel, SCHAR_MAX), SCHAR_MIN);
```

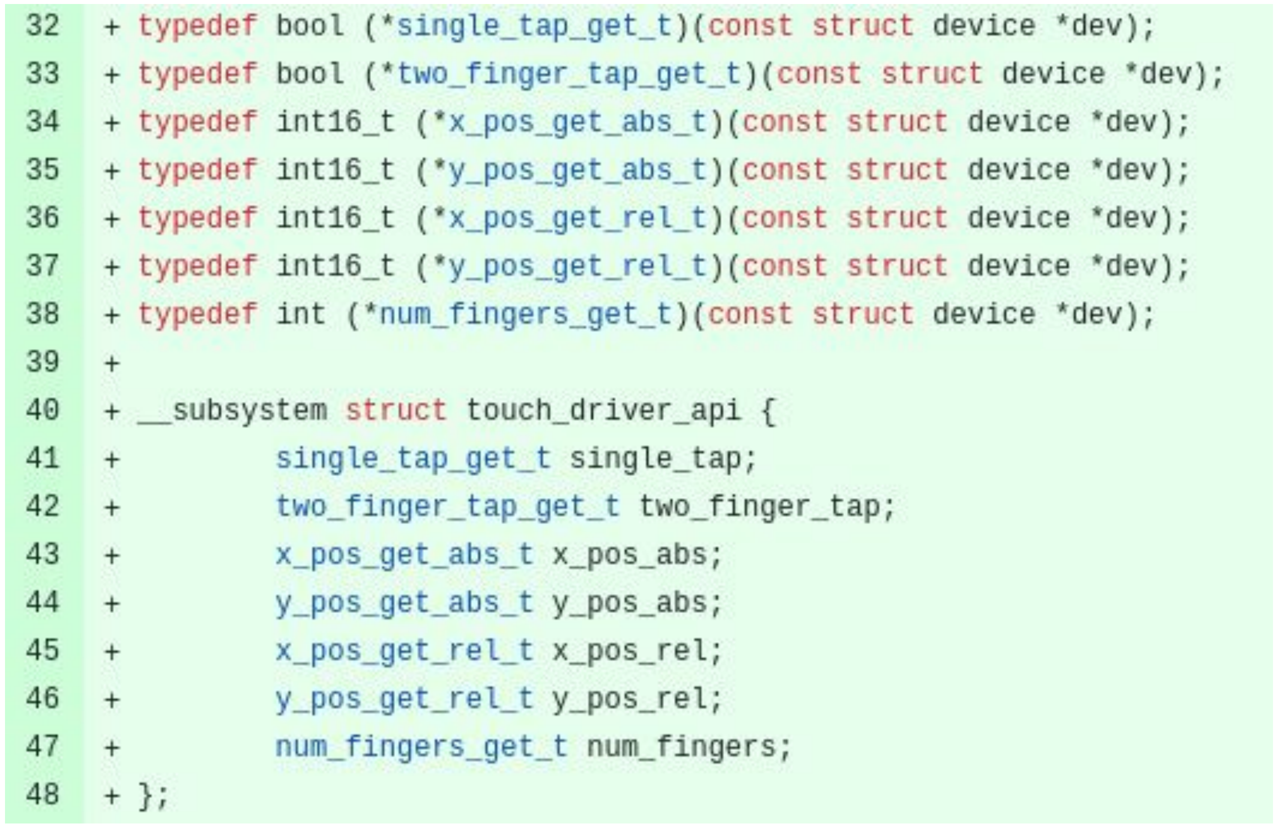
https://github.com/zephyrproject-rtos/zephyr/blob/60a20471b561a0a3a74b377991fe1976c2ea83d7/dts/bindings/sensor/nordic%2Cnrf-qdec.yaml
https://github.com/zephyrproject-rtos/zephyr/blob/60a20471b561a0a3a74b377991fe1976c2ea83d7/include/zephyr/drivers/sensor.h#L131
https://github.com/zephyrproject-rtos/zephyr/blob/60a20471b561a0a3a74b377991fe1976c2ea83d7/drivers/sensor/qdec_nrfx/qdec_nrfx.c#L104-L109
https://github.com/nrfconnect/sdk-nrf/blob/f3d31cc926d75ef5933259c2e998d34794a225bc/applications/nrf_desktop/src/hw_interface/wheel.c#L104-L106

Zephyr® Project
Developer Summit

Previous attempts

#EMBEDDEDOSSUMMIT

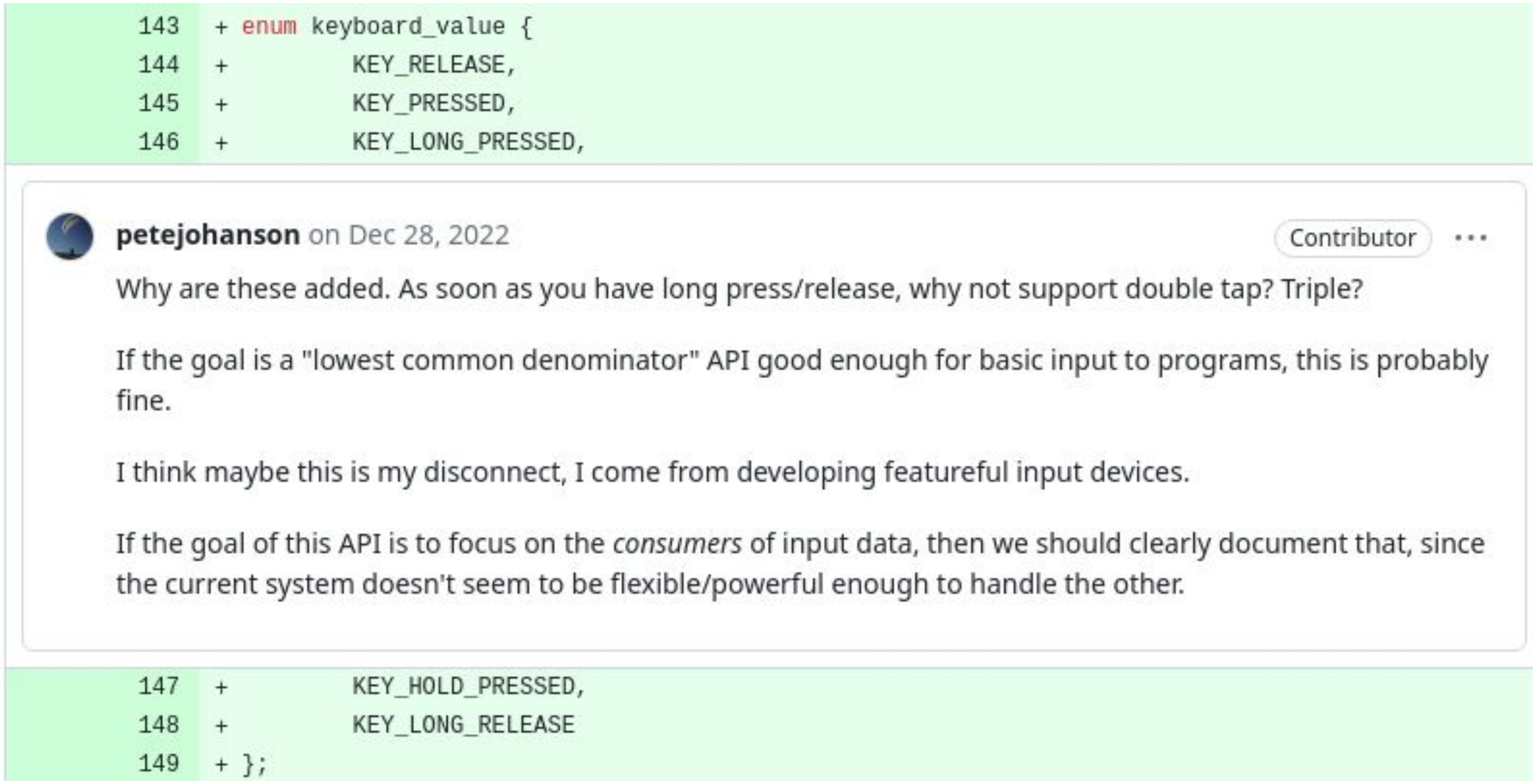
Device specific: touchscreen API



```
+ typedef bool (*single_tap_get_t)(const struct device *dev);
+ typedef bool (*two_finger_tap_get_t)(const struct device *dev);
+ typedef int16_t (*x_pos_get_abs_t)(const struct device *dev);
+ typedef int16_t (*y_pos_get_abs_t)(const struct device *dev);
+ typedef int16_t (*x_pos_get_rel_t)(const struct device *dev);
+ typedef int16_t (*y_pos_get_rel_t)(const struct device *dev);
+ typedef int (*num_fingers_get_t)(const struct device *dev);
+
+ __subsystem struct touch_driver_api {
+         single_tap_get_t single_tap;
+         two_finger_tap_get_t two_finger_tap;
+         x_pos_get_abs_t x_pos_abs;
+         y_pos_get_abs_t y_pos_abs;
+         x_pos_get_rel_t x_pos_rel;
+         y_pos_get_rel_t y_pos_rel;
+         num_fingers_get_t num_fingers;
+ };
```

https://github.com/zephyrproject-rtos/zephyr/pull/51493/files#diff-fe86c49654bc8c6aa24ca5f163198d74d797b62c2b7ae94a76fc41002e15d15dR265

Application Specific



```
+ enum keyboard_value {
+         KEY_RELEASE,
+         KEY_PRESSED,
+         KEY_LONG_PRESSED,
```

petejohanson on Dec 28, 2022 Contributor

Why are these added. As soon as you have long press/release, why not support double tap? Triple?

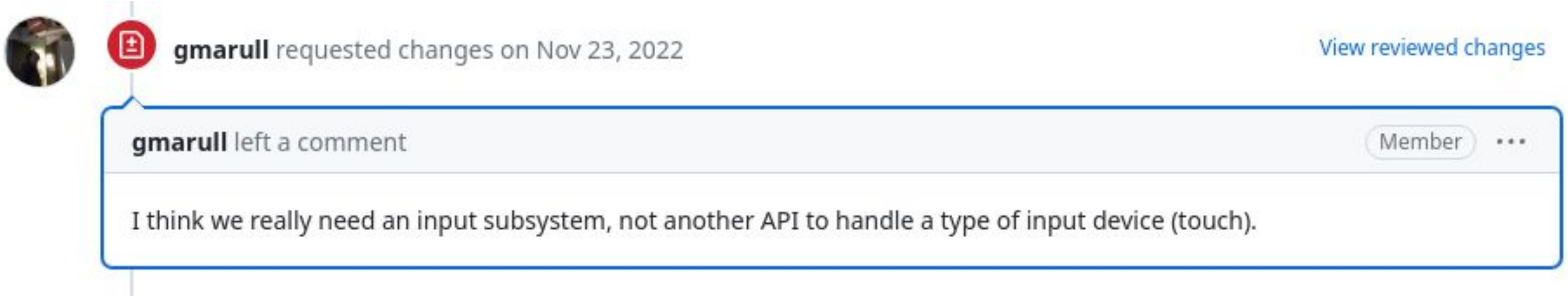
If the goal is a "lowest common denominator" API good enough for basic input to programs, this is probably fine.

I think maybe this is my disconnect, I come from developing featureful input devices.

If the goal of this API is to focus on the *consumers* of input data, then we should clearly document that, since the current system doesn't seem to be flexible/powerful enough to handle the other.

```
+         KEY_HOLD_PRESSED,
+         KEY_LONG_RELEASE
+ };
```

https://github.com/zephyrproject-rtos/zephyr/pull/48342/files



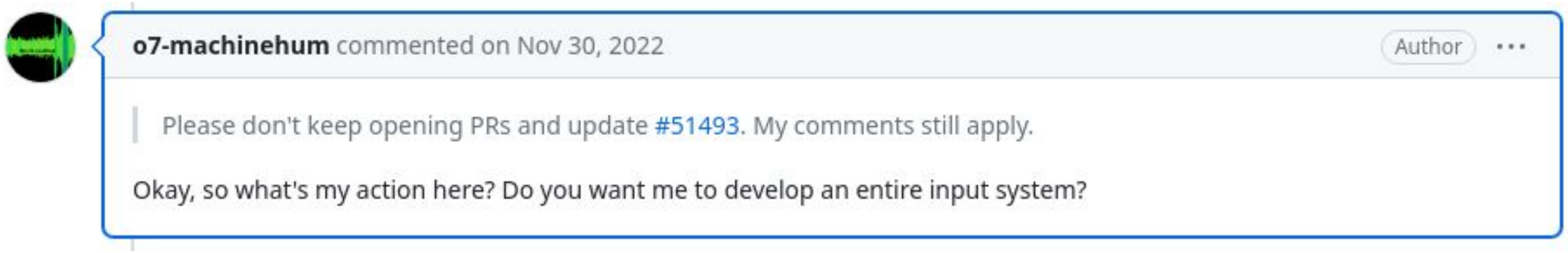
gmarull requested changes on Nov 23, 2022

View reviewed changes

gmarull left a comment

Member

I think we really need an input subsystem, not another API to handle a type of input device (touch).



o7-machinehum commented on Nov 30, 2022

Author

> Please don't keep opening PRs and update #51493. My comments still apply.

Okay, so what's my action here? Do you want me to develop an entire input system?

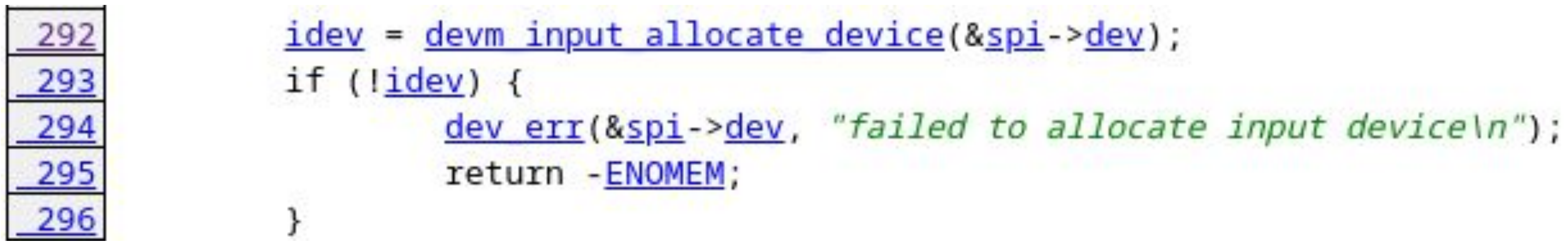
https://github.com/zephyrproject-rtos/zephyr/pull/51493#pullrequestreview-1191227398
https://github.com/zephyrproject-rtos/zephyr/pull/52649#issuecomment-1332508482

Zephyr® Project
Developer Summit

Linux Input Events

#EMBEDDEDOSSUMMIT

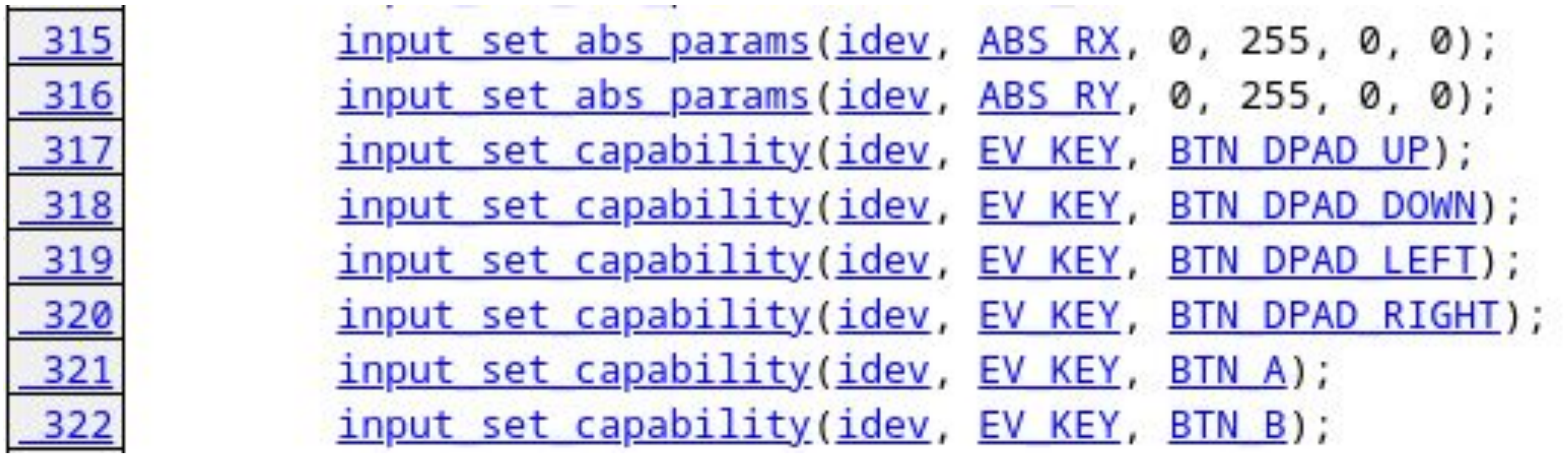
Allocate your device



```
idev = devm_input_allocate_device(&spi->dev);
if (!idev) {
        dev_err(&spi->dev, "failed to allocate input device\n");
        return -ENOMEM;
}
```

https://lxr.missinglinkelectronics.com/linux/drivers/input/joystick/psxpad-spi.c#L292

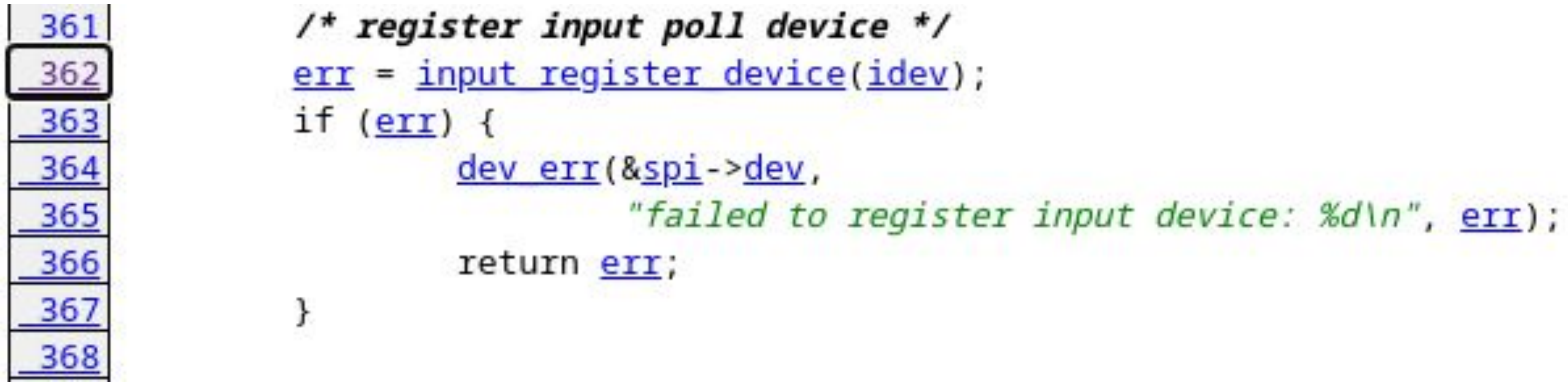
Set capabilities



```
input_set_abs_params(idev, ABS_RX, 0, 255, 0, 0);
input_set_abs_params(idev, ABS_RY, 0, 255, 0, 0);
input_set_capability(idev, EV_KEY, BTN_DPAD_UP);
input_set_capability(idev, EV_KEY, BTN_DPAD_DOWN);
input_set_capability(idev, EV_KEY, BTN_DPAD_LEFT);
input_set_capability(idev, EV_KEY, BTN_DPAD_RIGHT);
input_set_capability(idev, EV_KEY, BTN_A);
input_set_capability(idev, EV_KEY, BTN_B);
```

https://lxr.missinglinkelectronics.com/linux/drivers/input/joystick/psxpad-spi.c#L313

Register a device

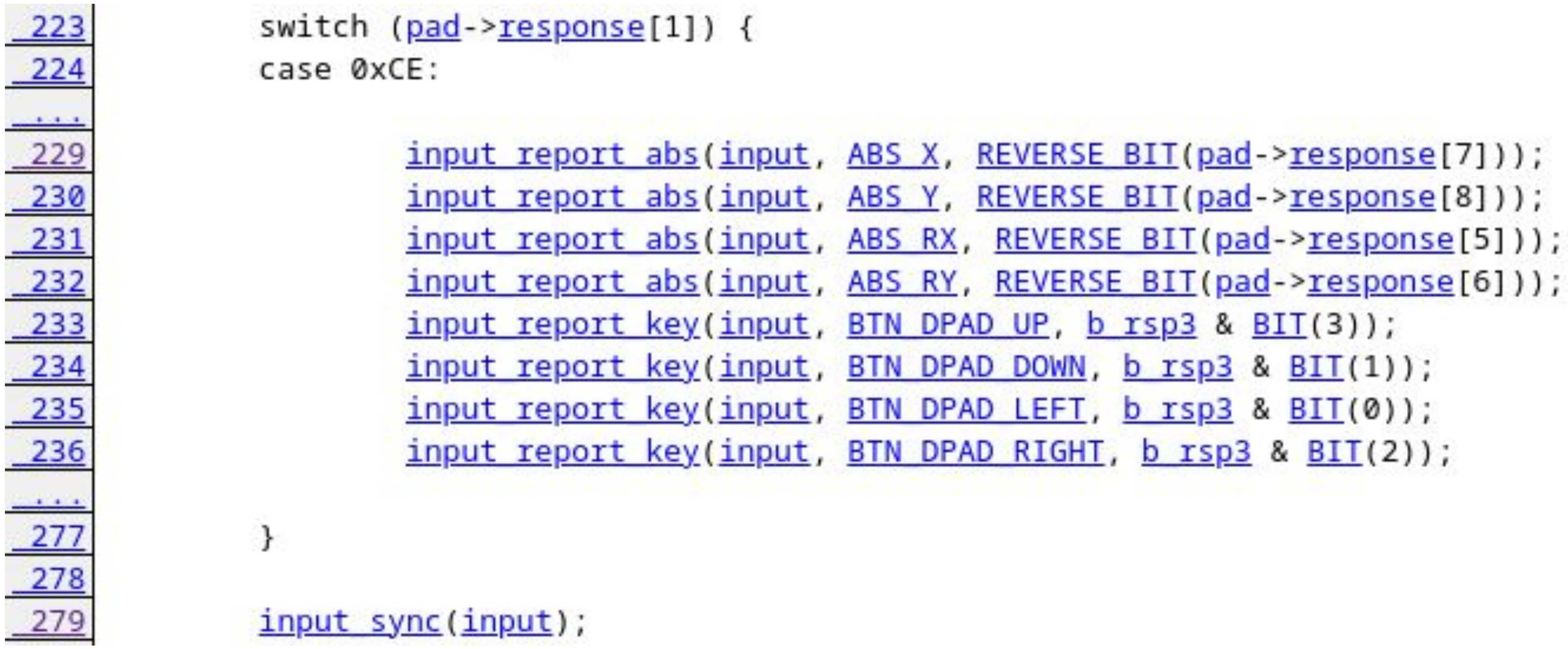


```
/* register input poll device */
err = input_register_device(idev);
if (err) {
        dev_err(&spi->dev,
                "failed to register input device: %d\n", err);
        return err;
}
```

https://lxr.missinglinkelectronics.com/linux/drivers/input/joystick/psxpad-spi.c#L362

Zephyr® Project
Developer Summit

Send reports and sync

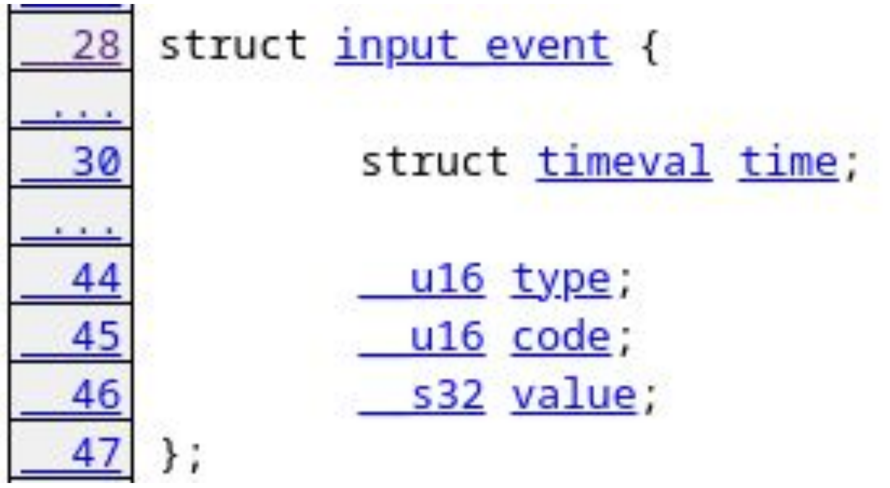


```
	switch (pad->response[1]) {
	case 0xCE:
...
		input_report_abs(input, ABS_X, REVERSE_BIT(pad->response[7]));
		input_report_abs(input, ABS_Y, REVERSE_BIT(pad->response[8]));
		input_report_abs(input, ABS_RX, REVERSE_BIT(pad->response[5]));
		input_report_abs(input, ABS_RY, REVERSE_BIT(pad->response[6]));
		input_report_key(input, BTN_DPAD_UP, b_rsp3 & BIT(3));
		input_report_key(input, BTN_DPAD_DOWN, b_rsp3 & BIT(1));
		input_report_key(input, BTN_DPAD_LEFT, b_rsp3 & BIT(0));
		input_report_key(input, BTN_DPAD_RIGHT, b_rsp3 & BIT(2));
...
	}

	input_sync(input);
```

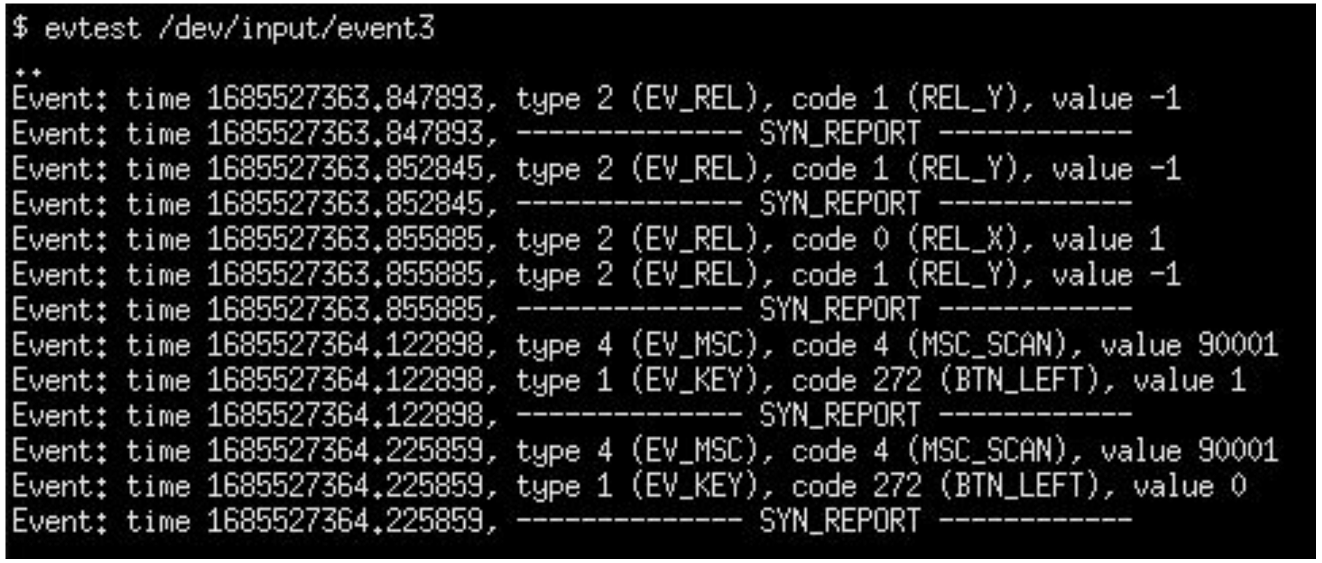
https://lxr.missinglinkelectronics.com/linux/drivers/input/joystick/psxpad-spi.c#L229

Zephyr® Project
Developer Summit

Receive events from /dev/input/event<n>



```
struct input_event {

        struct timeval time;

        __u16 type;
        __u16 code;
        __s32 value;
};
```



```
$ evtest /dev/input/event3
..
Event: time 1685527363.847893, type 2 (EV_REL), code 1 (REL_Y), value -1
Event: time 1685527363.847893, -------------- SYN_REPORT ------------
Event: time 1685527363.852845, type 2 (EV_REL), code 1 (REL_Y), value -1
Event: time 1685527363.852845, -------------- SYN_REPORT ------------
Event: time 1685527363.855885, type 2 (EV_REL), code 0 (REL_X), value 1
Event: time 1685527363.855885, type 2 (EV_REL), code 1 (REL_Y), value -1
Event: time 1685527363.855885, -------------- SYN_REPORT ------------
Event: time 1685527364.122898, type 4 (EV_MSC), code 4 (MSC_SCAN), value 90001
Event: time 1685527364.122898, type 1 (EV_KEY), code 272 (BTN_LEFT), value 1
Event: time 1685527364.122898, -------------- SYN_REPORT ------------
Event: time 1685527364.225859, type 4 (EV_MSC), code 4 (MSC_SCAN), value 90001
Event: time 1685527364.225859, type 1 (EV_KEY), code 272 (BTN_LEFT), value 0
Event: time 1685527364.225859, -------------- SYN_REPORT ------------
```

https://lxr.missinglinkelectronics.com/linux/include/uapi/linux/input.h#L28

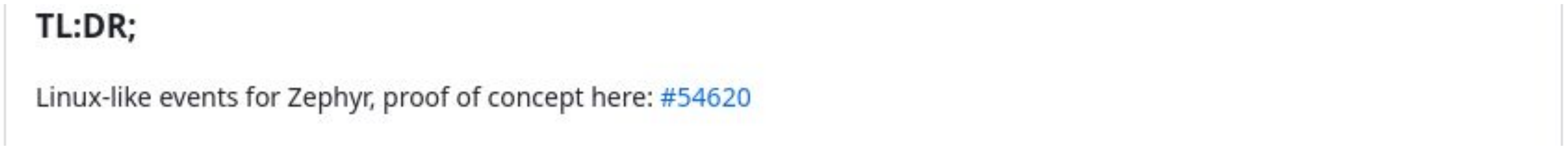
Zephyr® Project
Developer Summit

Zephyr Implementation

#EMBEDDEDOSSUMMIT

Zephyr® Project
Developer Summit

Zephyr Input Subsystem proposal #54622



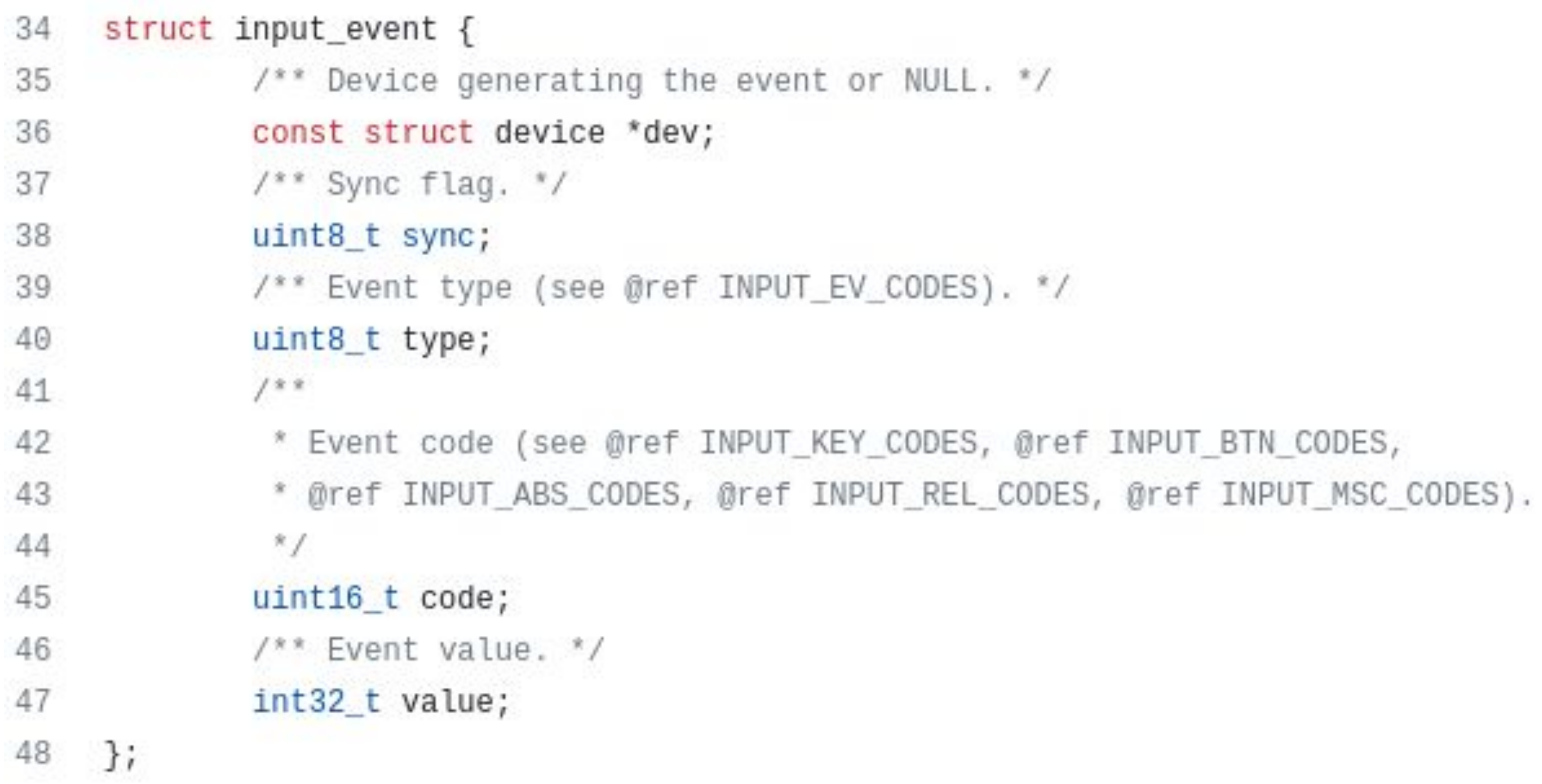
TL:DR;

Linux-like events for Zephyr, proof of concept here: #54620

Requirements:
- Support a many-to-many model, register listeners at build time
- Support events from any type of input device without special functions
- Backwards compatible with kscan
- Provide a framework for extensibility

https://github.com/zephyrproject-rtos/zephyr/issues/54622

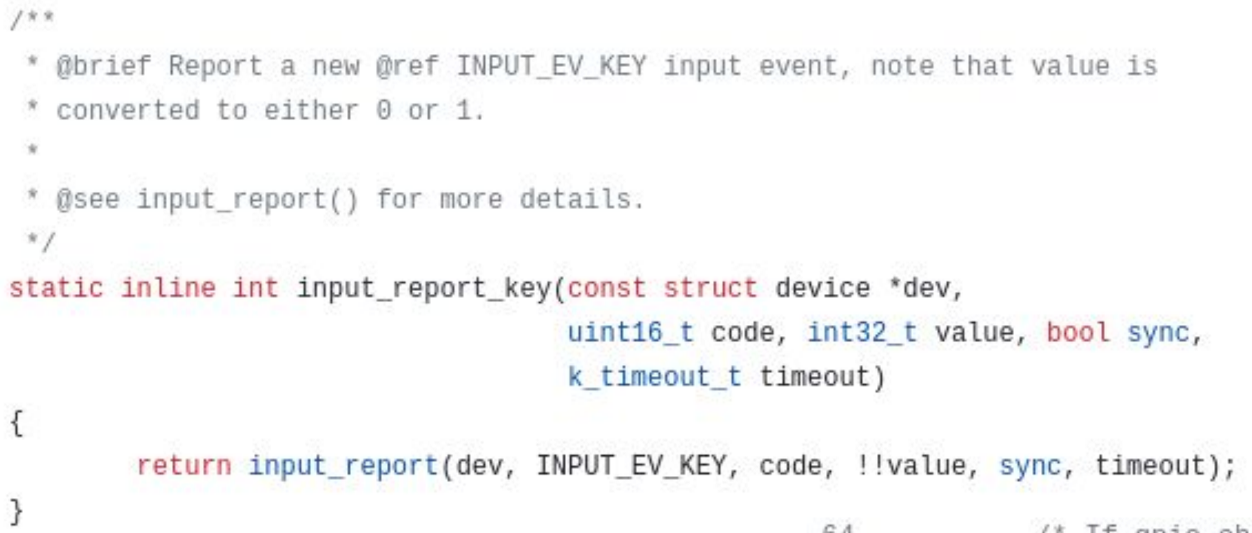
Input Event structure



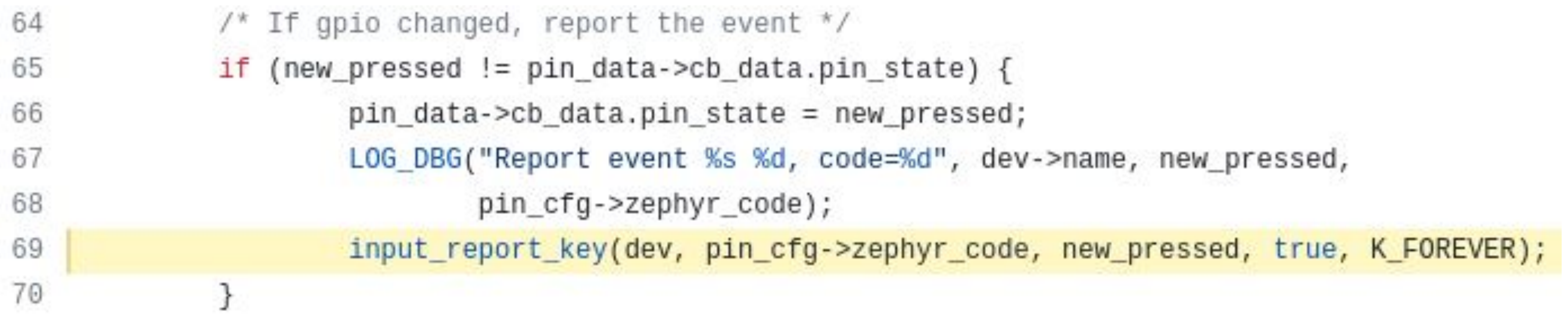
```c
struct input_event {
        /** Device generating the event or NULL. */
        const struct device *dev;
        /** Sync flag. */
        uint8_t sync;
        /** Event type (see @ref INPUT_EV_CODES). */
        uint8_t type;
        /**
         * Event code (see @ref INPUT_KEY_CODES, @ref INPUT_BTN_CODES,
         * @ref INPUT_ABS_CODES, @ref INPUT_REL_CODES, @ref INPUT_MSC_CODES).
         */
        uint16_t code;
        /** Event value. */
        int32_t value;
};
```

https://github.com/zephyrproject-rtos/zephyr/blob/main/include/zephyr/input/input.h#L34-L48

Driver API: gpio-keys



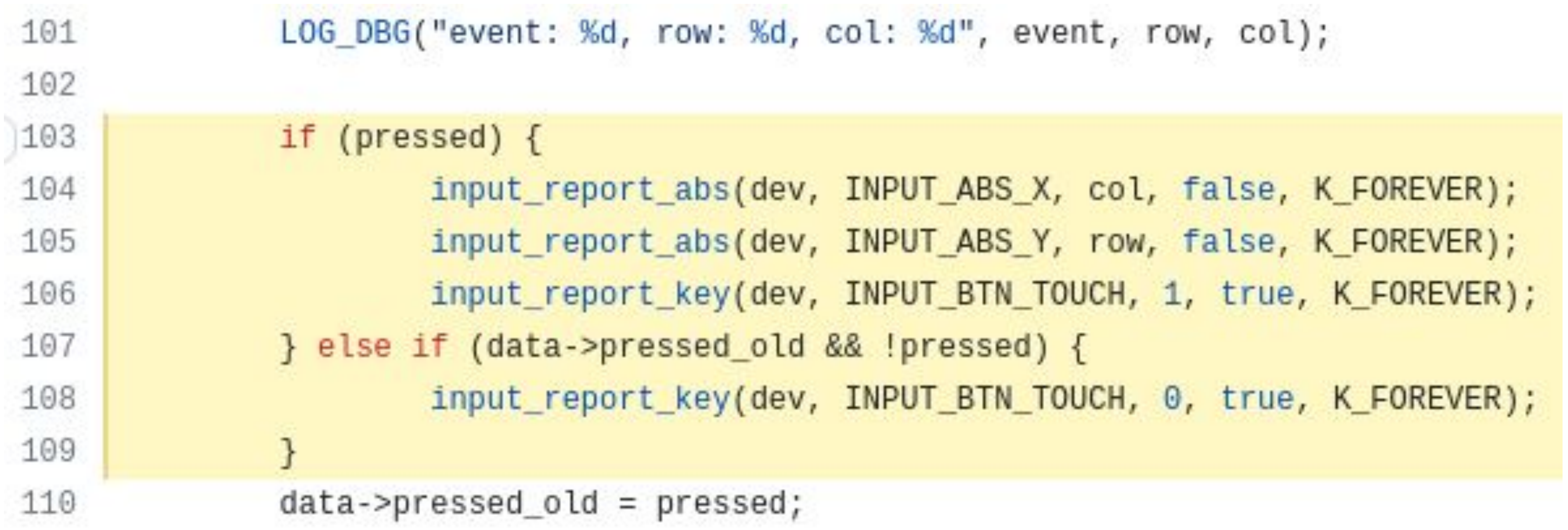
```
/**
 * @brief Report a new @ref INPUT_EV_KEY input event, note that value is
 * converted to either 0 or 1.
 *
 * @see input_report() for more details.
 */
static inline int input_report_key(const struct device *dev,
                                   uint16_t code, int32_t value, bool sync,
                                   k_timeout_t timeout)
{
        return input_report(dev, INPUT_EV_KEY, code, !!value, sync, timeout);
}
```



```
        /* If gpio changed, report the event */
        if (new_pressed != pin_data->cb_data.pin_state) {
                pin_data->cb_data.pin_state = new_pressed;
                LOG_DBG("Report event %s %d, code=%d", dev->name, new_pressed,
                        pin_cfg->zephyr_code);
                input_report_key(dev, pin_cfg->zephyr_code, new_pressed, true, K_FOREVER);
        }
```

https://github.com/zephyrproject-rtos/zephyr/blob/main/include/zephyr/input/input.h
https://github.com/zephyrproject-rtos/zephyr/blob/main/drivers/input/input_gpio_keys.c#L69

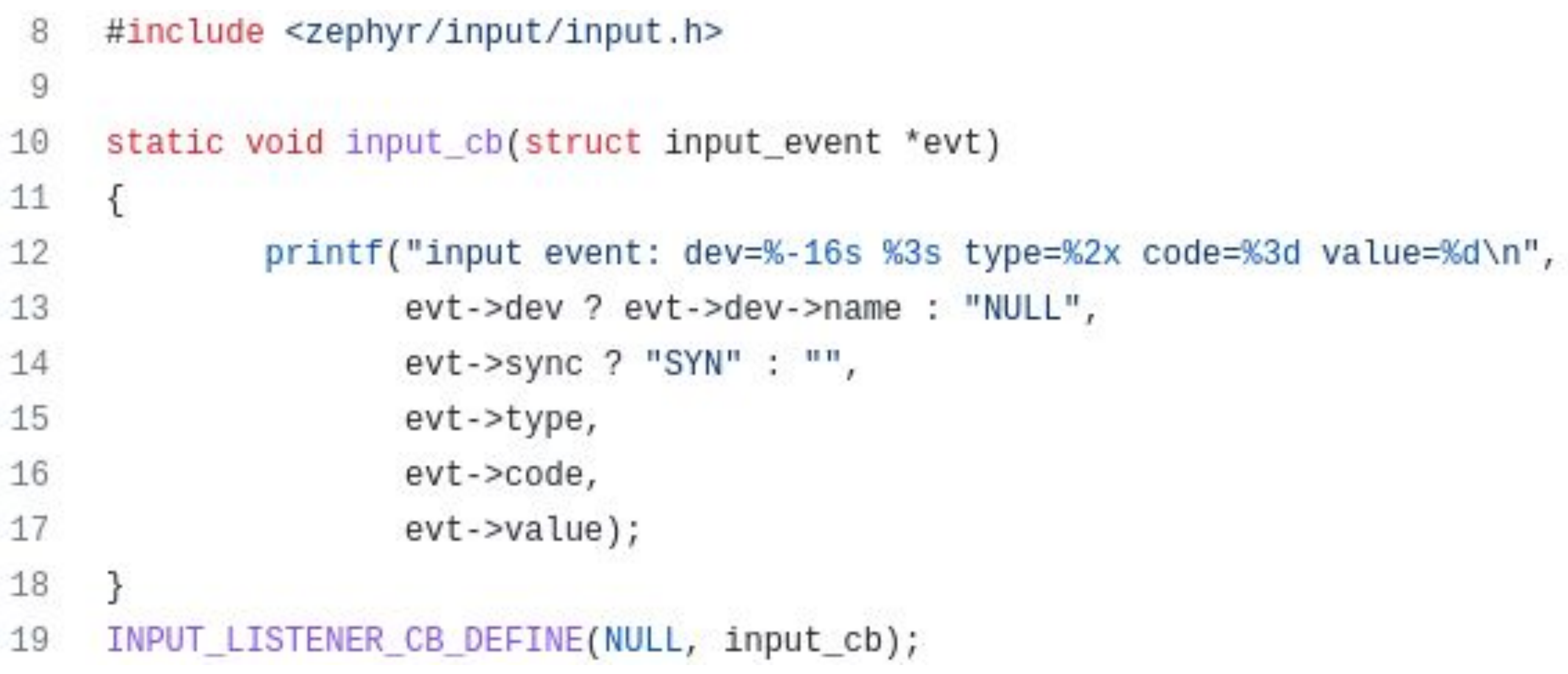
Driver API: touchscreen



```c
101         LOG_DBG("event: %d, row: %d, col: %d", event, row, col);
102 
103         if (pressed) {
104                 input_report_abs(dev, INPUT_ABS_X, col, false, K_FOREVER);
105                 input_report_abs(dev, INPUT_ABS_Y, row, false, K_FOREVER);
106                 input_report_key(dev, INPUT_BTN_TOUCH, 1, true, K_FOREVER);
107         } else if (data->pressed_old && !pressed) {
108                 input_report_key(dev, INPUT_BTN_TOUCH, 0, true, K_FOREVER);
109         }
110         data->pressed_old = pressed;
```

https://github.com/zephyrproject-rtos/zephyr/blob/main/drivers/input/input_ft5336.c#L103-L109

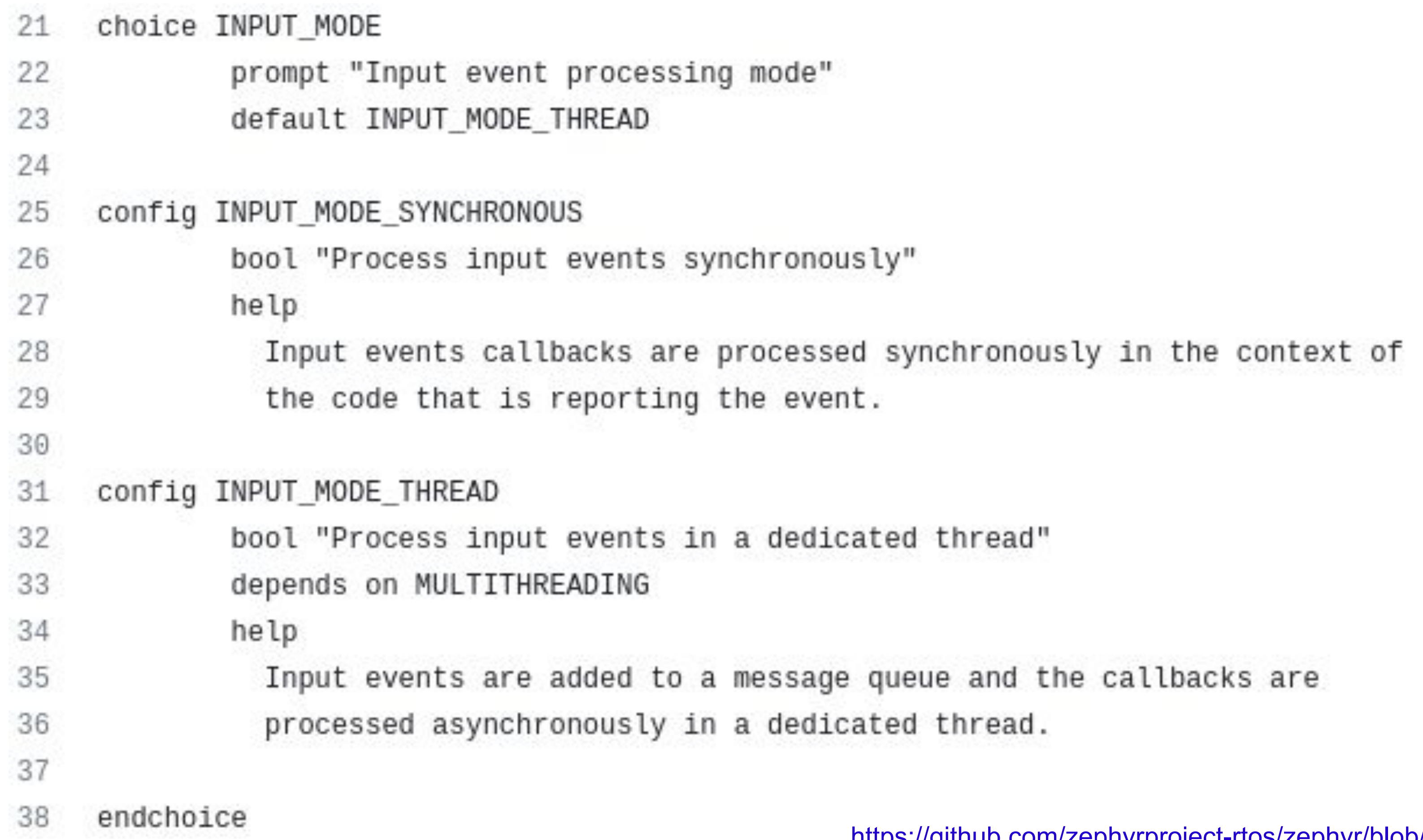
Application API



```c
#include <zephyr/input/input.h>

static void input_cb(struct input_event *evt)
{
        printf("input event: dev=%-16s %3s type=%2x code=%3d value=%d\n",
               evt->dev ? evt->dev->name : "NULL",
               evt->sync ? "SYN" : "",
               evt->type,
               evt->code,
               evt->value);
}
INPUT_LISTENER_CB_DEFINE(NULL, input_cb);
```

https://github.com/zephyrproject-rtos/zephyr/blob/main/samples/subsys/input/input_dump/src/main.c#L3

Modes of operation

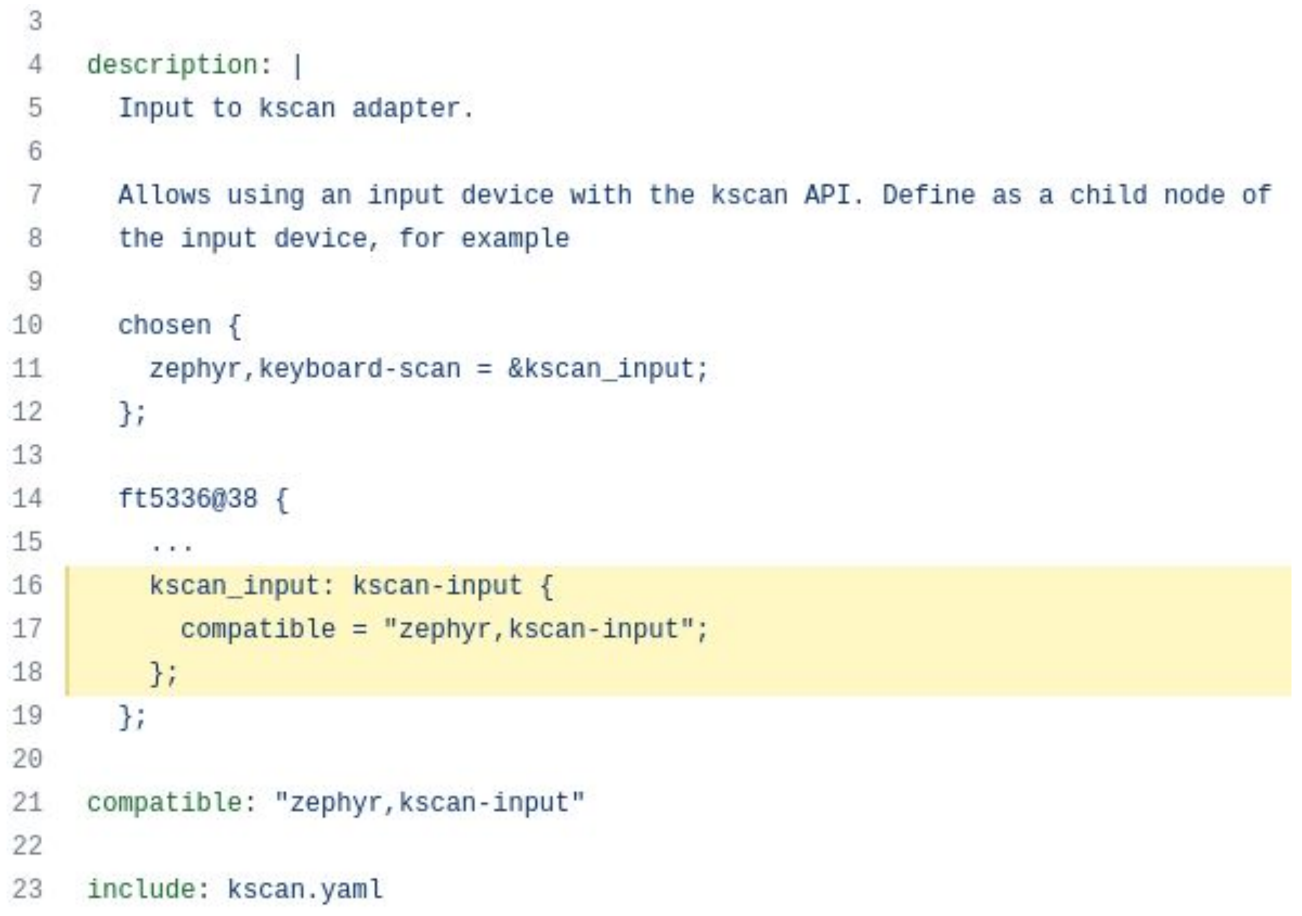


```
21  choice INPUT_MODE
22          prompt "Input event processing mode"
23          default INPUT_MODE_THREAD
24
25  config INPUT_MODE_SYNCHRONOUS
26          bool "Process input events synchronously"
27          help
28            Input events callbacks are processed synchronously in the context of
29            the code that is reporting the event.
30
31  config INPUT_MODE_THREAD
32          bool "Process input events in a dedicated thread"
33          depends on MULTITHREADING
34          help
35            Input events are added to a message queue and the callbacks are
36            processed asynchronously in a dedicated thread.
37
38  endchoice
```

https://github.com/zephyrproject-rtos/zephyr/blob/main/subsys/input/Kconfig#L21-L38

Zephyr® Project
Developer Summit

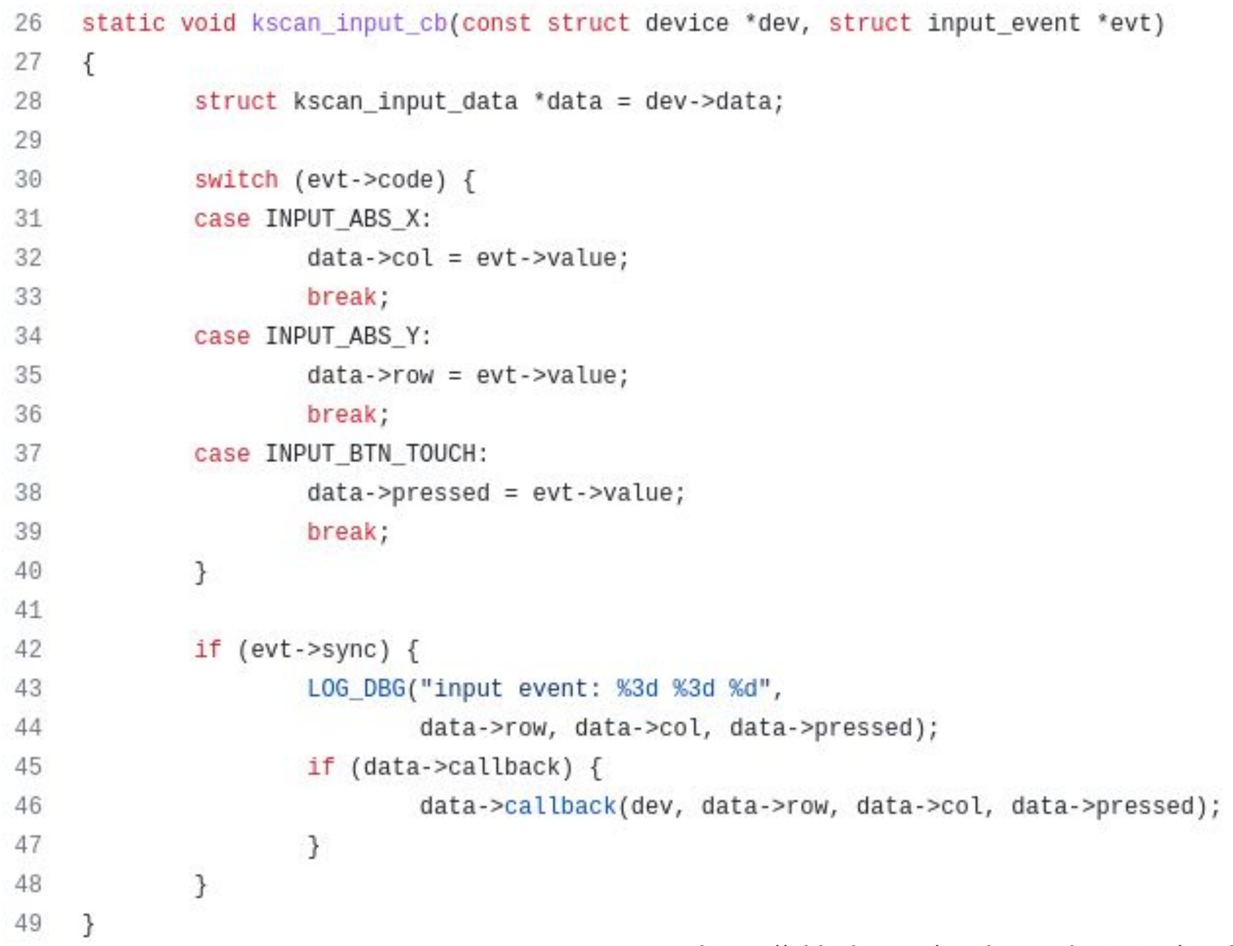
Kscan compatibility



```
description: |
  Input to kscan adapter.

  Allows using an input device with the kscan API. Define as a child node of
  the input device, for example

  chosen {
    zephyr,keyboard-scan = &kscan_input;
  };

  ft5336@38 {
    ...
    kscan_input: kscan-input {
      compatible = "zephyr,kscan-input";
    };
  };

compatible: "zephyr,kscan-input"

include: kscan.yaml
```

https://github.com/zephyrproject-rtos/zephyr/blob/main/dts/bindings/kscan/zephyr%2Ckscan-input.yaml

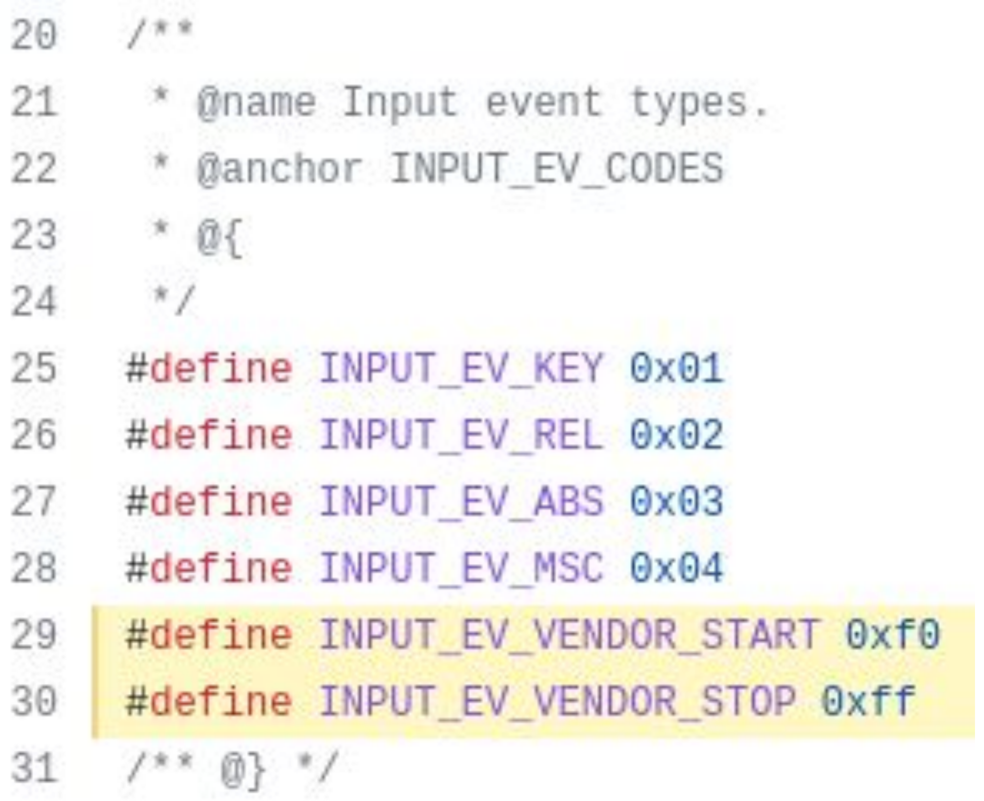
Kscan compatibility



```c
static void kscan_input_cb(const struct device *dev, struct input_event *evt)
{
	struct kscan_input_data *data = dev->data;

	switch (evt->code) {
	case INPUT_ABS_X:
		data->col = evt->value;
		break;
	case INPUT_ABS_Y:
		data->row = evt->value;
		break;
	case INPUT_BTN_TOUCH:
		data->pressed = evt->value;
		break;
	}

	if (evt->sync) {
		LOG_DBG("input event: %3d %3d %d",
			data->row, data->col, data->pressed);
		if (data->callback) {
			data->callback(dev, data->row, data->col, data->pressed);
		}
	}
}
```

https://github.com/zephyrproject-rtos/zephyr/blob/main/drivers/kscan/kscan_input.c#L26-L49

Extensibility

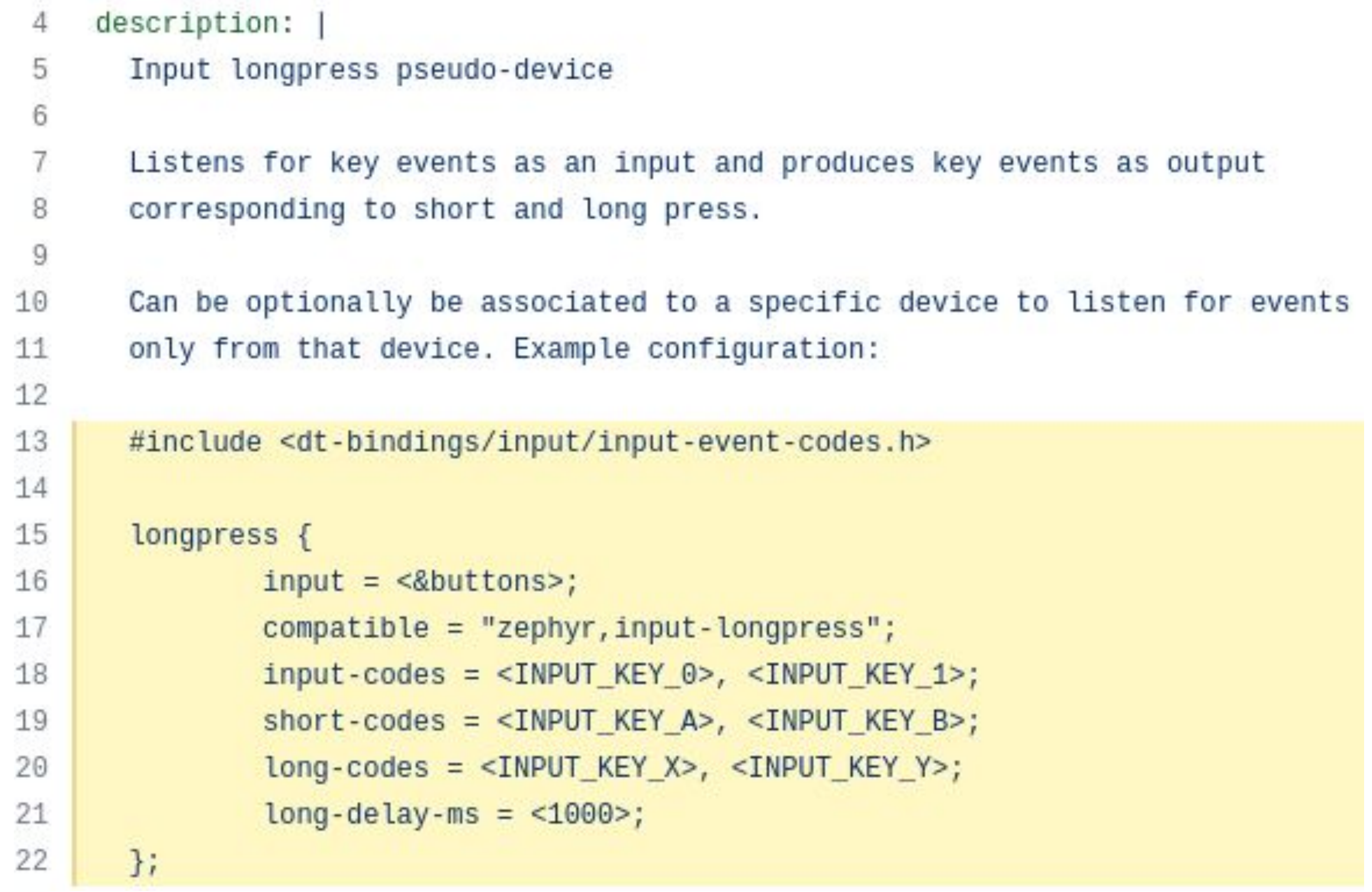


```
/**
 * @name Input event types.
 * @anchor INPUT_EV_CODES
 * @{
 */
#define INPUT_EV_KEY 0x01
#define INPUT_EV_REL 0x02
#define INPUT_EV_ABS 0x03
#define INPUT_EV_MSC 0x04
#define INPUT_EV_VENDOR_START 0xf0
#define INPUT_EV_VENDOR_STOP 0xff
/** @} */
```

https://github.com/zephyrproject-rtos/zephyr/blob/main/include/zephyr/dt-bindings/input/input-event-codes.h#L29-L30

Zephyr® Project
Developer Summit

Filters and event reprocessing



```
 4  description: |
 5    Input longpress pseudo-device
 6
 7    Listens for key events as an input and produces key events as output
 8    corresponding to short and long press.
 9
10    Can be optionally be associated to a specific device to listen for events
11    only from that device. Example configuration:
12
13    #include <dt-bindings/input/input-event-codes.h>
14
15    longpress {
16            input = <&buttons>;
17            compatible = "zephyr,input-longpress";
18            input-codes = <INPUT_KEY_0>, <INPUT_KEY_1>;
19            short-codes = <INPUT_KEY_A>, <INPUT_KEY_B>;
20            long-codes = <INPUT_KEY_X>, <INPUT_KEY_Y>;
21            long-delay-ms = <1000>;
22    };
```

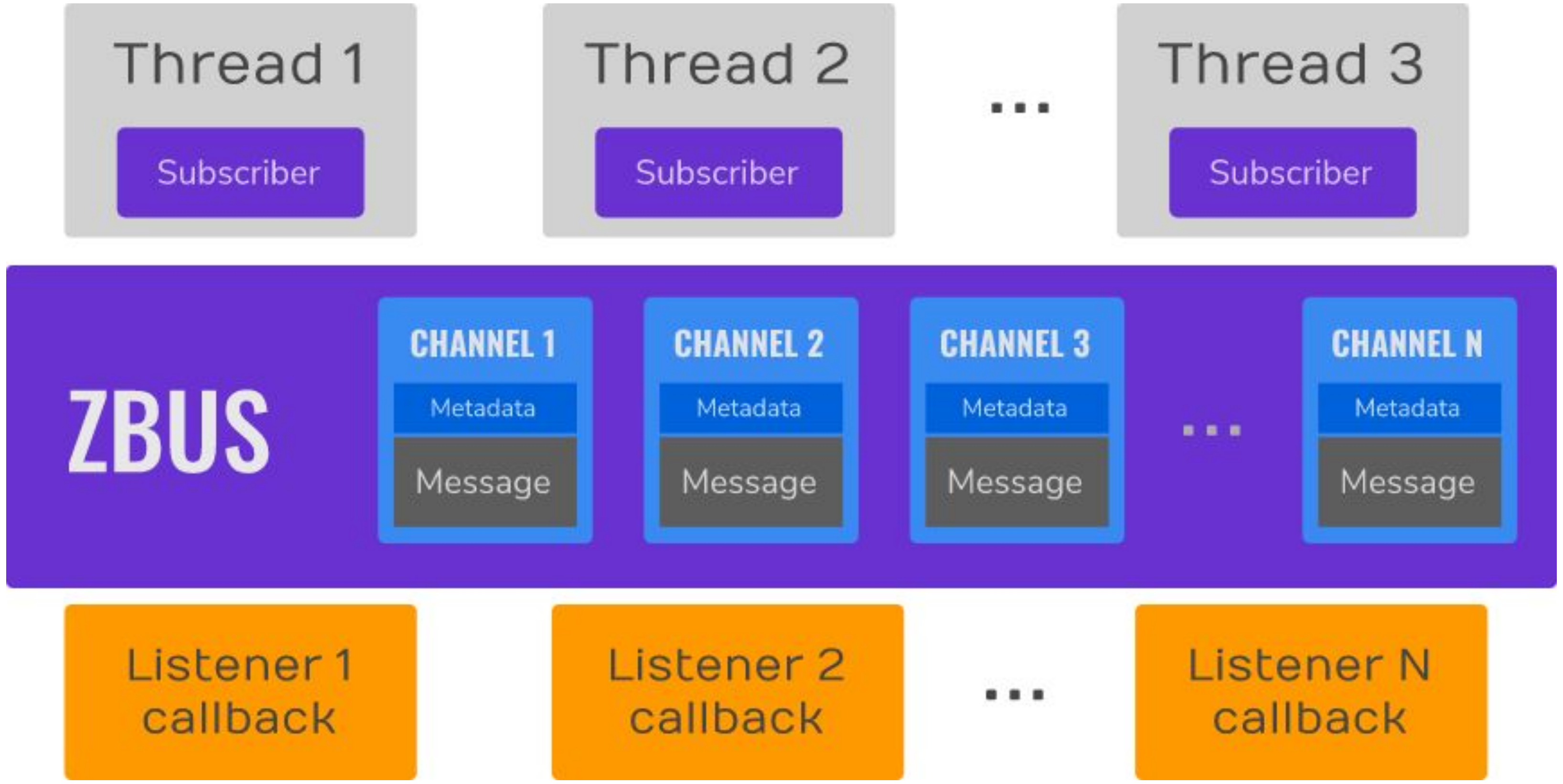
https://github.com/zephyrproject-rtos/zephyr/blob/main/dts/bindings/input/zephyr%2Cinput-longpress.yaml#L13-L22

Zephyr® Project
Developer Summit

Event Systems

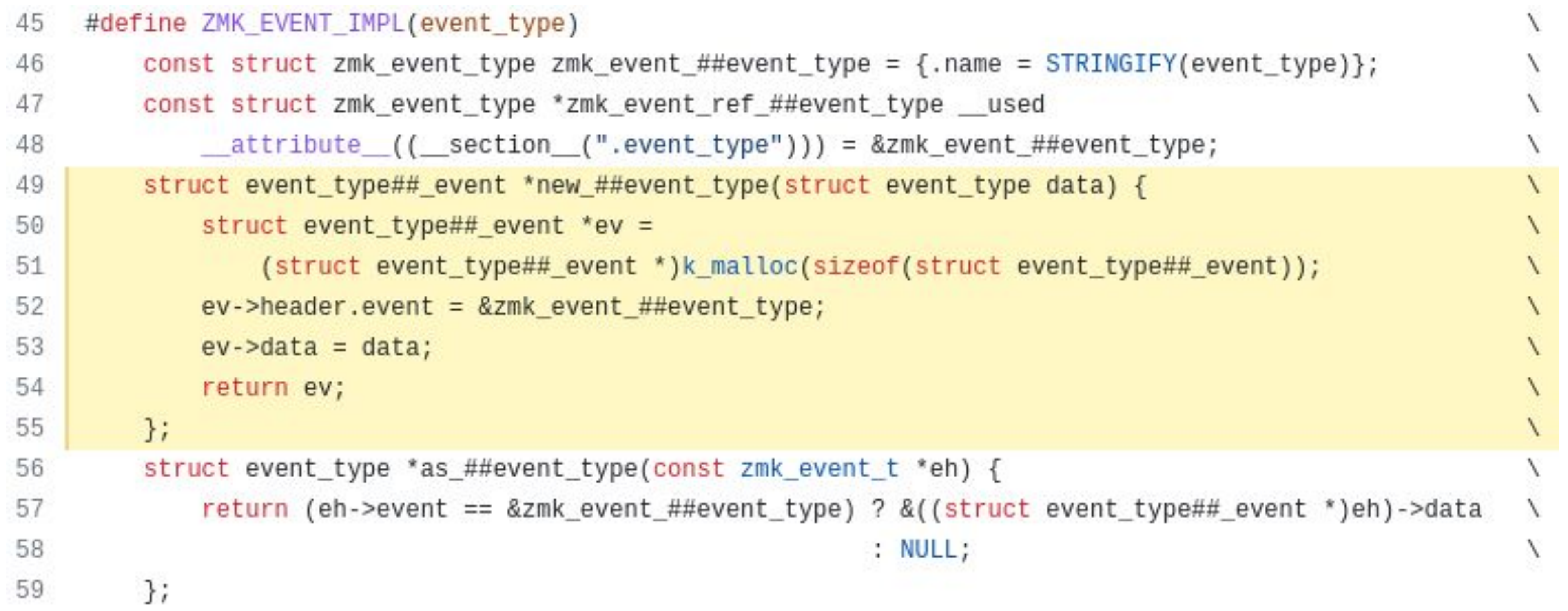
#EMBEDDEDOSSUMMIT

Zbus



https://docs.zephyrproject.org/latest/services/zbus/index.html

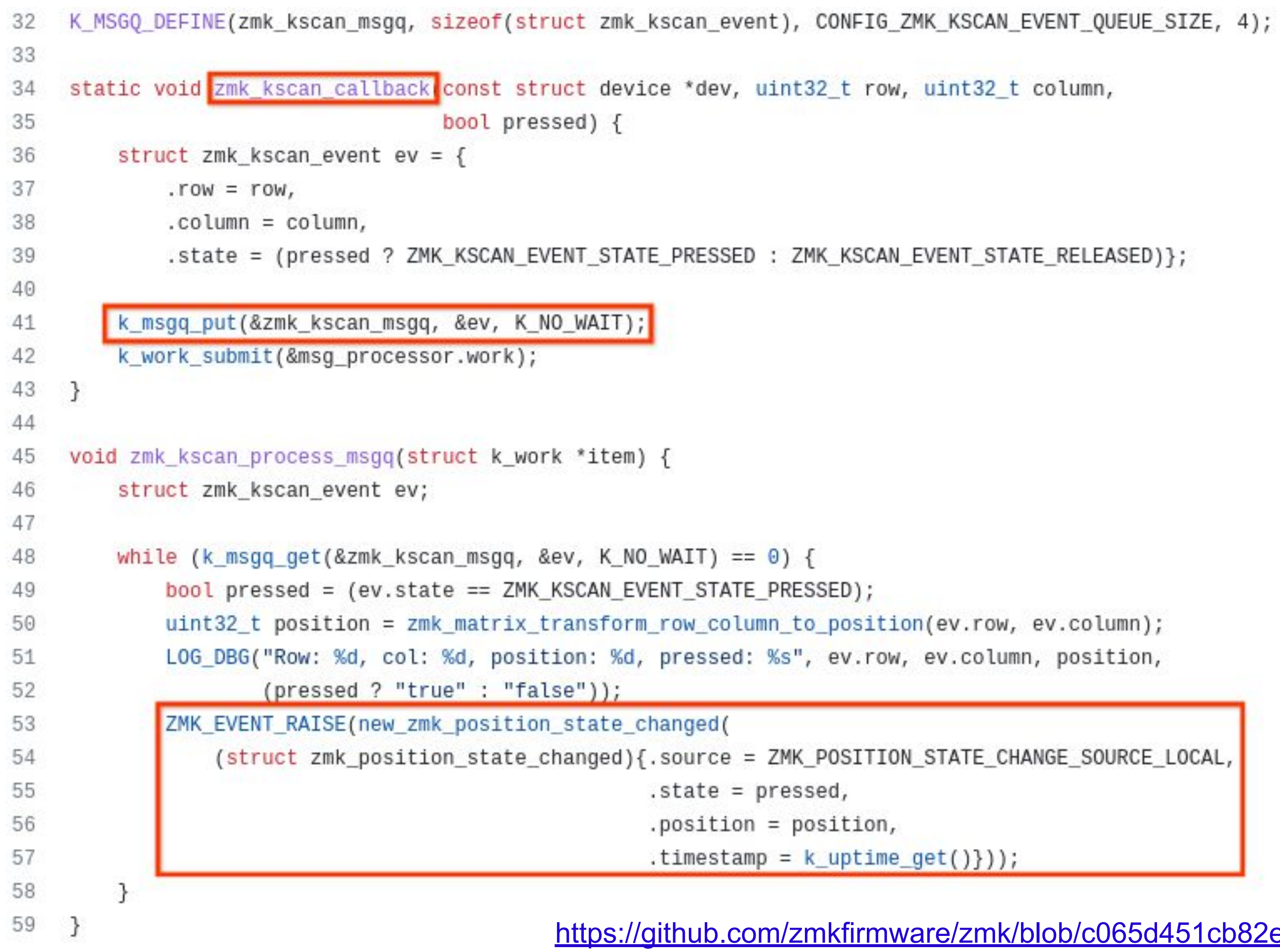
ZMK event system



```c
45 #define ZMK_EVENT_IMPL(event_type)                                                                 \
46     const struct zmk_event_type zmk_event_##event_type = {.name = STRINGIFY(event_type)};          \
47     const struct zmk_event_type *zmk_event_ref_##event_type __used                                 \
48         __attribute__((__section__(".event_type"))) = &zmk_event_##event_type;                     \
49     struct event_type##_event *new_##event_type(struct event_type data) {                          \
50         struct event_type##_event *ev =                                                            \
51             (struct event_type##_event *)k_malloc(sizeof(struct event_type##_event));              \
52         ev->header.event = &zmk_event_##event_type;                                                \
53         ev->data = data;                                                                           \
54         return ev;                                                                                 \
55     };                                                                                             \
56     struct event_type *as_##event_type(const zmk_event_t *eh) {                                    \
57         return (eh->event == &zmk_event_##event_type) ? &((struct event_type##_event *)eh)->data   \
58                                                       : NULL;                                      \
59     };
```

https://github.com/zmkfirmware/zmk/blob/c065d451cb82e2dceda4efdb2991aeeeef1cdbbf/app/include/zmk/event_manager.h#L49-L55

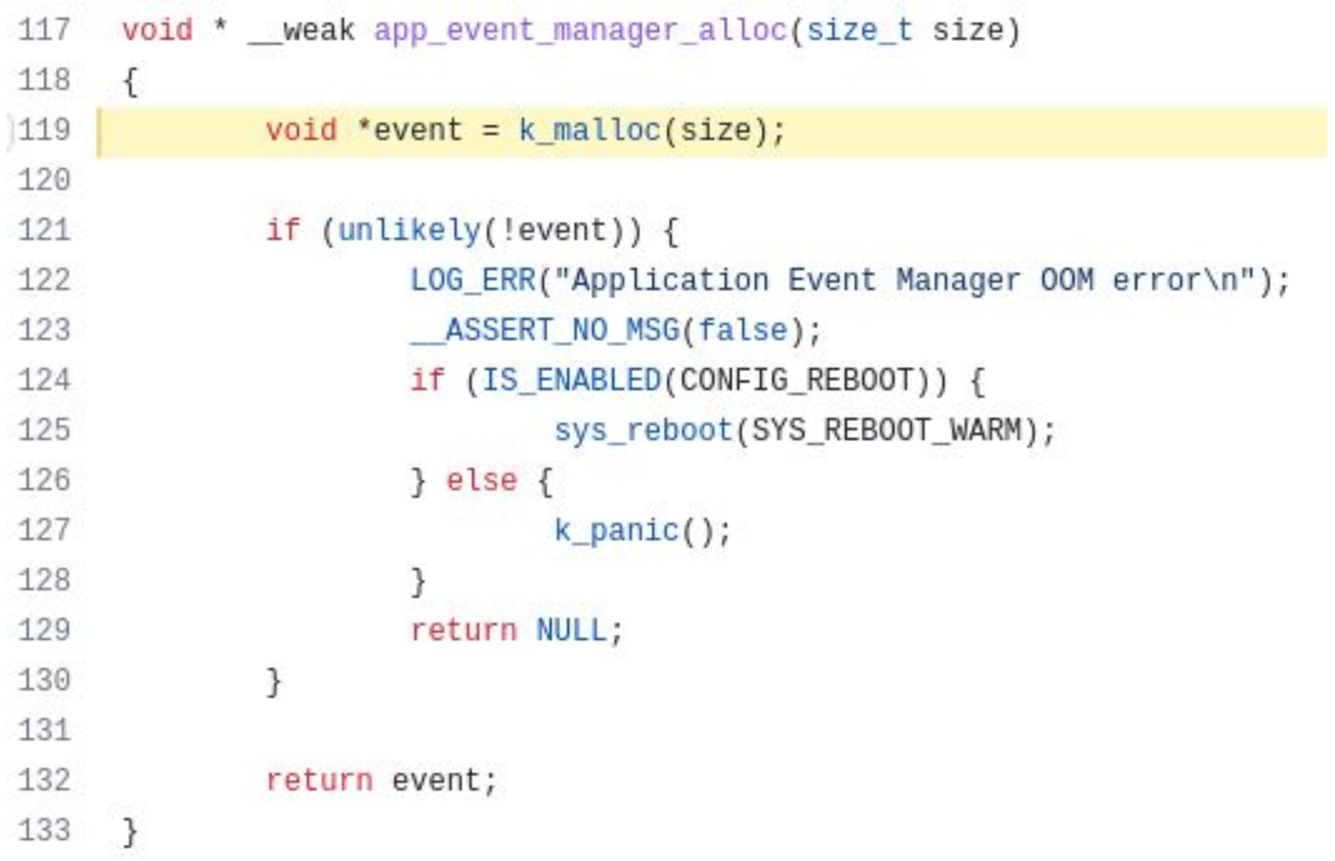
ZMK event processing



```c
K_MSGQ_DEFINE(zmk_kscan_msgq, sizeof(struct zmk_kscan_event), CONFIG_ZMK_KSCAN_EVENT_QUEUE_SIZE, 4);

static void zmk_kscan_callback(const struct device *dev, uint32_t row, uint32_t column,
                               bool pressed) {
    struct zmk_kscan_event ev = {
        .row = row,
        .column = column,
        .state = (pressed ? ZMK_KSCAN_EVENT_STATE_PRESSED : ZMK_KSCAN_EVENT_STATE_RELEASED)};

    k_msgq_put(&zmk_kscan_msgq, &ev, K_NO_WAIT);
    k_work_submit(&msg_processor.work);
}

void zmk_kscan_process_msgq(struct k_work *item) {
    struct zmk_kscan_event ev;

    while (k_msgq_get(&zmk_kscan_msgq, &ev, K_NO_WAIT) == 0) {
        bool pressed = (ev.state == ZMK_KSCAN_EVENT_STATE_PRESSED);
        uint32_t position = zmk_matrix_transform_row_column_to_position(ev.row, ev.column);
        LOG_DBG("Row: %d, col: %d, position: %d, pressed: %s", ev.row, ev.column, position,
                (pressed ? "true" : "false"));
        ZMK_EVENT_RAISE(new_zmk_position_state_changed(
            (struct zmk_position_state_changed){.source = ZMK_POSITION_STATE_CHANGE_SOURCE_LOCAL,
                                                .state = pressed,
                                                .position = position,
                                                .timestamp = k_uptime_get()}));
    }
}
```

https://github.com/zmkfirmware/zmk/blob/c065d451cb82e2dceda4efdb2991aeeeef1cdbbf/app/src/kscan.c

NCS event manager



```c
void * __weak app_event_manager_alloc(size_t size)
{
	void *event = k_malloc(size);

	if (unlikely(!event)) {
		LOG_ERR("Application Event Manager OOM error\n");
		__ASSERT_NO_MSG(false);
		if (IS_ENABLED(CONFIG_REBOOT)) {
			sys_reboot(SYS_REBOOT_WARM);
		} else {
			k_panic();
		}
		return NULL;
	}

	return event;
}
```

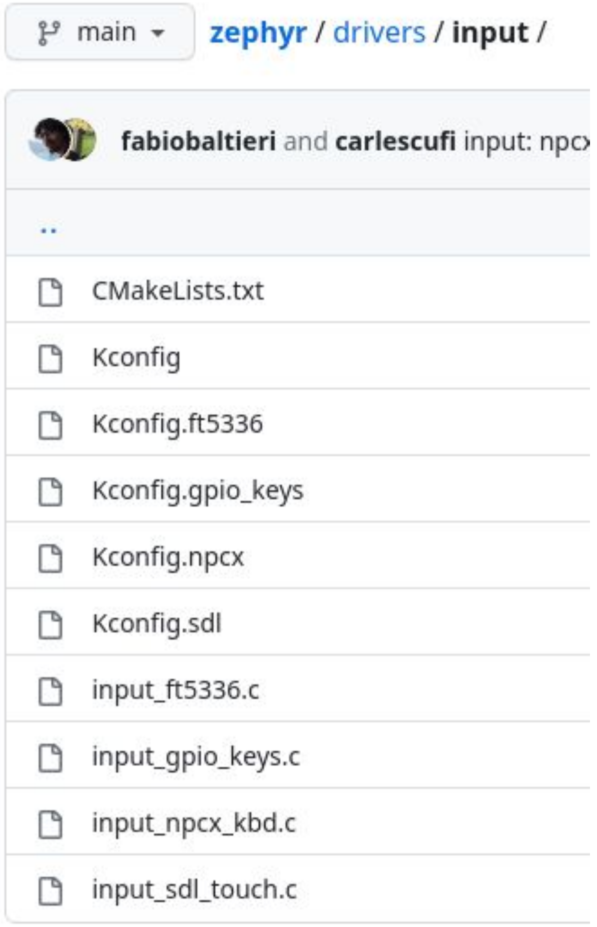
https://github.com/nrfconnect/sdk-nrf/blob/d0bade499df9fd37d6a87f60d33f052f6f1acf10/subsys/app_event_manager/app_event_manager.c#L119

Zephyr® Project
Developer Summit

What now?

#EMBEDDEDOSSUMMIT

State of the Input subsystem



main zephyr / drivers / input /

fabiobaltieri and carlescufi input: npc

..

CMakeLists.txt

Kconfig

Kconfig.ft5336

Kconfig.gpio_keys

Kconfig.npcx

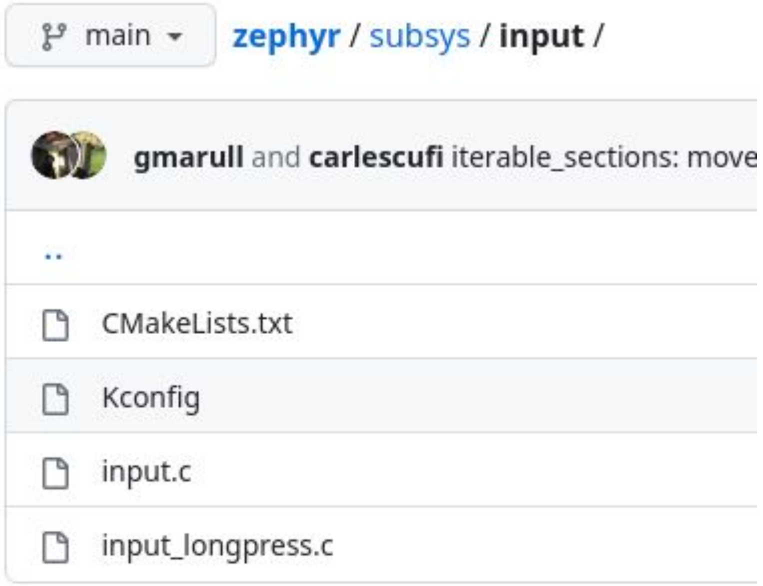
Kconfig.sdl

input_ft5336.c

input_gpio_keys.c

input_npcx_kbd.c

input_sdl_touch.c



main zephyr / subsys / input /

gmarull and carlescufi iterable_sections: move

..

CMakeLists.txt

Kconfig

input.c

input_longpress.c

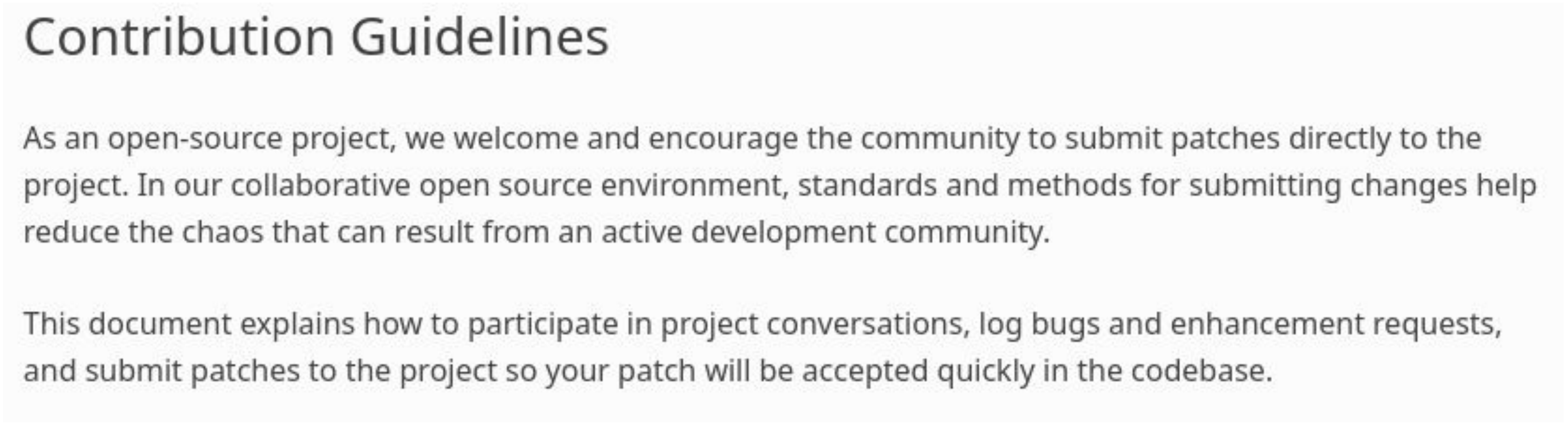
https://github.com/zephyrproject-rtos/zephyr/tree/main/drivers/input
https://github.com/zephyrproject-rtos/zephyr/tree/main/subsys/input

Next

- Finish converting the existing drivers
- Zbus integration
- Add some common code for keyboard scanning matrixes
- Write native sinks for LVGL
- Deprecate Kscan

Zephyr® Project
Developer Summit

Send patches!



Contribution Guidelines

As an open-source project, we welcome and encourage the community to submit patches directly to the project. In our collaborative open source environment, standards and methods for submitting changes help reduce the chaos that can result from an active development community.

This document explains how to participate in project conversations, log bugs and enhancement requests, and submit patches to the project so your patch will be accepted quickly in the codebase.

https://docs.zephyrproject.org/latest/contribute/guidelines.html

Zephyr® Project
Developer Summit

Questions