

# Demo Design: UIO IRQ

Draft v3.0

Alex He ([ahe@xilinx.com](mailto:ahe@xilinx.com))

The design show how to implement the UIO module in project.

Git: [https://gitenterprise.xilinx.com/AlexHe/UIO\\_Linux\\_Demo](https://gitenterprise.xilinx.com/AlexHe/UIO_Linux_Demo)

## Prerequisites

- Vivado 2017.2
- Petalinux 2017.2
- ZCU102 EVB final v1.0

## STEP

### Create 4 interrupt input pins of zcu102 EVB

Use the GPIO\_DIP\_SW[0-7]:SW13 of User I/O (Refer to **UG1182-ZCU102 Evaluation Board**)

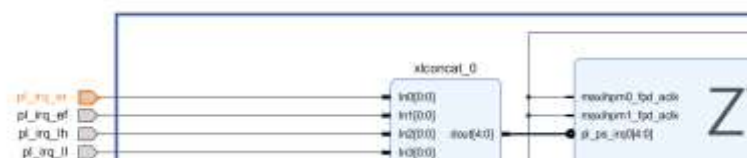
GPIO DIP SW (Active High)			
AN14	GPIO_DIP_SW0	LVC MOS33	SW13.8

Table 3-33: XCZU9EG U1 to GPIO Connections (Cont'd)

XCZU9EG (U1) Pin	Schematic Net Name	I/O Standard	Device
AP14	GPIO_DIP_SW1	LVC MOS33	SW13.7
AM14	GPIO_DIP_SW2	LVC MOS33	SW13.6
AN13	GPIO_DIP_SW3	LVC MOS33	SW13.5
AN12	GPIO_DIP_SW4	LVC MOS33	SW13.4
AP12	GPIO_DIP_SW5	LVC MOS33	SW13.3
AL13	GPIO_DIP_SW6	LVC MOS33	SW13.2
AK13	GPIO_DIP_SW7	LVC MOS33	SW13.1

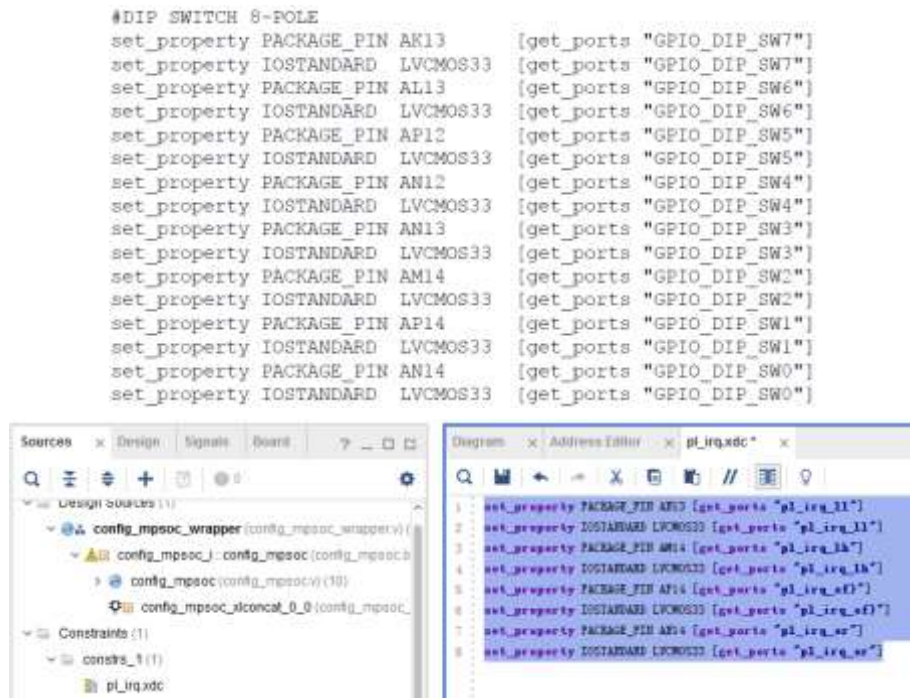
Connect the 4 ports to pl\_ps\_irq0 through xlconcat.

- |                          |   |           |                      |
|--------------------------|---|-----------|----------------------|
| In0 – Edge/Raising Edge  | ⇔ | pl_irq_er | (AN14, SW13.8, DIP0) |
| In1 – Edge/ Falling Edge | ⇔ | pl_irq_ef | (AP14, SW13.7, DIP1) |
| In2 – Level/Active High  | ⇔ | pl_irq_lh | (AM14, SW13.6, DIP2) |
| In3 - Level/Active Low   | ⇔ | pl_irq_ll | (AN13, SW13.5, DIP3) |

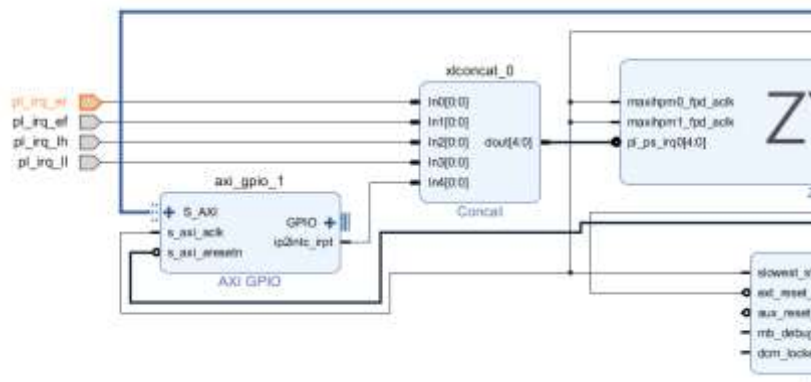


XILINX CONFIDENTIAL

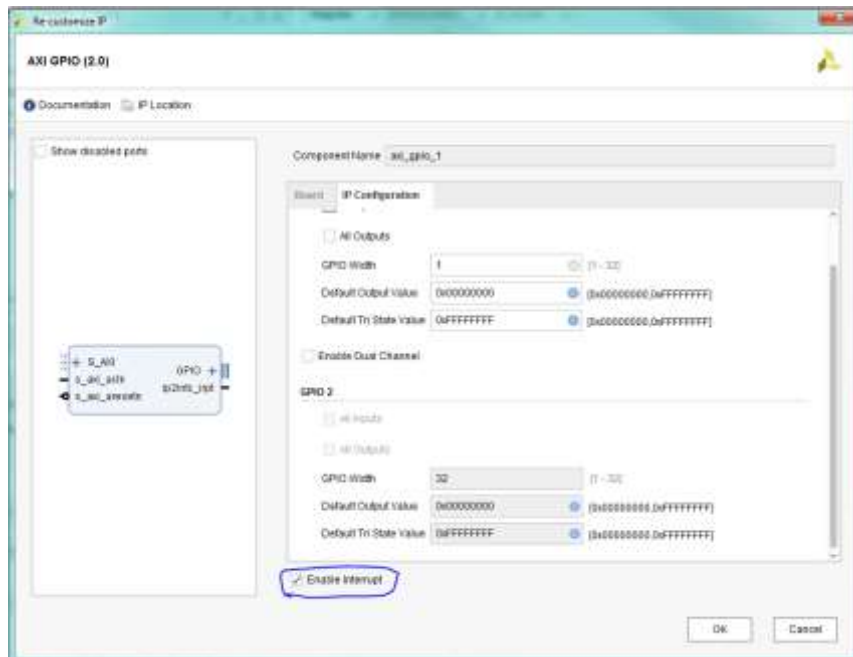
Add constrains (refer to **UG1182**, ZCU102 BOARD Constraints File Listing)



Add a GPIO interrupt source



GPIO IP setting with enable interrupt



Connect the GPIO to PL LEDs (led\_8bits) on board.

Then the 5 pl\_ps\_irq[4:0] was created in sequence as **Table 13-1 of UG1085**.

Table 13-1: System Interrupts (Cont'd)

IRQ Name	IRQ Number	GICPx_IRQ Bits	Description
USB0_OTG	101	GICP2 [5]	USB 0 OTG mode
USB1_Endpoint	102:105	GICP2 [6:9]	Bulk transfer, isochronous transfer, controller interrupt, control transfer.
USB1_OTG	106	GICP2 [10]	USB 1 OTG mode
USB0_Wakeup	107	GICP2 [11]	USB 0 controller to wake-up PMU.
USB1_Wakeup	108	GICP2 [12]	USB 1 controller to wake-up PMU.
LPD_DMA	109:116	GICP2 [13:20]	Eight DMA channels: channels 0 to 7.
CSU	117	GICP2 [21]	Configuration and security unit.
CSU_DMA	118	GICP2 [22]	CSU DMA controller.
eFuse	119	GICP2 [23]	eFuse interrupt.
XPPIU	120	GICP2 [24]	Peripheral protection unit in LPD.
<u>PL_PS_Group0</u>	<u>121:128</u>	GICP2 [25:31] GICP3[0]	PL to PS interrupt signals 0 to 7.

So the IRQ number for the 5 input are here.

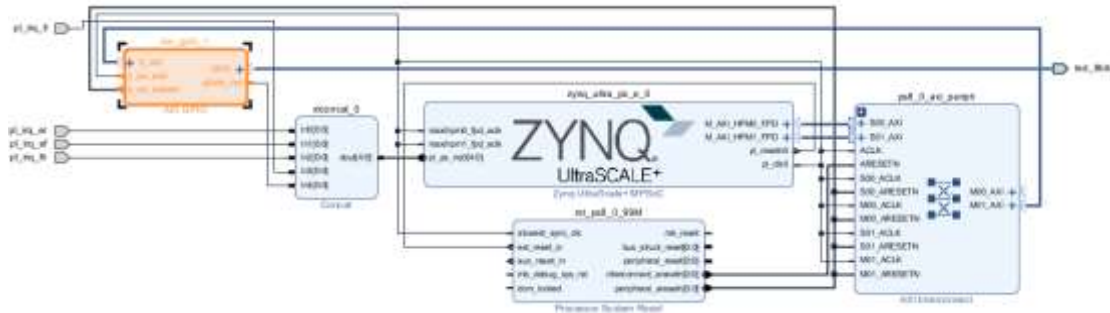
IRQ source	IRQ number	Board Info
pl_irq_er	121	SW13.8, DIP0
pl_irq_ef	122	SW13.7, DIP1
pl_irq_lh	123	SW13.6, DIP2
pl_irq_ll	124	SW13.5, DIP3
axi_gpio_1	125	

Address Map



Then use the IPI flow to create the bitstream and export the Hardware(HDF)

The design diagram is here.



## Create the petalinux project with the HDF

```
$petalinux-create -t project --template zynqMP -n zcu102-pl2ps_irq
```

```
$cd ./zcu102-pl2ps_irq
```

```
$petalinux-config --get-hw-description <path of HDF>
```

```
$petalinux-config -c kernel
```

Enable UIO\_PDRV\_GENIRQ driver

```
CONFIG_UIO=y
```

```
# CONFIG_UIO_CIF is not set
```

```
CONFIG_UIO_PDRV_GENIRQ=y
```

```
$petalinux-build -c device-tree
```

The pl.dtsi is created in ./components/plnx\_workspace/device-tree-generation/ as below.

```

* CAUTION: This file is automatically generated by xilinx.
* Version:
* Today is: Tue Aug 8 18:49:16 2017
*/

/ {
    amba_pl: amba_pl@0 {
        #address-cells = <2>;
        #size-cells = <2>;
        compatible = "xilinx,amba-pl";
        ranges;
        axi_gpio_1: gpio@0010000 {
            #gpio-cells = <2>;
            #interrupt-cells = <2>;
            compatible = "xilinx,xps-gpio-1.00.a";
            gpio-controller;
            interrupt-controller;
            interrupt-parent = <0>;
            interrupts = <0 89 4>;
            reg = <0x0 0x0010000 0x0 0x10000>;
            xlnx,all-inputs = <0x0>;
            xlnx,all-inputs-2 = <0x1>;
            xlnx,all-outputs = <0x1>;
            xlnx,all-outputs-2 = <0x0>;
            xlnx,dout-default = <0x00000000>;
            xlnx,dout-default-2 = <0x00000000>;
            xlnx,gpio-width = <0x0>;
            xlnx,gpio2-width = <0x5>;
            xlnx,interrupt-present = <0x1>;
            xlnx,ss-dual = <0x0>;
            xlnx,tri-default = <0xffffffff>;
            xlnx,tri-default-2 = <0xffffffff>;
        };
        pdc_ctrl_ip1: PERIPHERAL@ff380080 {
            compatible = "xlnx,PERIPHERAL-1.0";
            reg = <0x0 0xff380080 0x0 0x00000>;
        };
        pdc_message_buffers: PERIPHERAL@ff900000 {
            compatible = "xlnx,PERIPHERAL-1.0";
            reg = <0x0 0xff900000 0x0 0x10000>;
        };
    };
};

```

The DTS should be modified for

1. Refine the axi\_gpio\_1 node as UIO
  - a) Change the interrupts value from 89 to 93
  - b) Change the compatible to "generic-uio" to use UIO\_PDRV\_GENIRQ driver.
2. Add UIO node for the each 4 DIP ports (to trigger pl\_ps\_irq0[3:0])
  - a) Set the interrupts value from 89 to 92
  - b) Set compatible to "generic-uio"

\*The interrupt value of node in DTS has an offset 32 of the IRQ number (Table 13-1: **System Interrupts, UG1085**).

E.g.

The value 89 should add 32 to get the real hardware IRQ number which is 121, i.e. the first interrupt number of pl\_ps\_group0.

\*DTS interrupts binding

```

/*----- Interrupts -----*/
/*
 * Interrupt-parent = <0>;
 * Interrupts = <0 89 4>; /* (decoded to vsc #)
 */
};

/*# two cells
-----
The #interrupt-cells property is set to 2 and the first cell defines the
index of the interrupt within the controller, while the second cell is used
to specify any of the following flags:
- bits[3:0] trigger type and level flags
  1 = low-to-high edge triggered
  2 = high-to-low edge triggered
  4 = active high level-sensitive
  8 = active low level-sensitive
*/

```

Set the UIO nodes in ./project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi

```

#include "system-conf.dtsi"
/ {
    amba_pl00 {
        #address-cells = <0x2>;
        #size-cells = <0x2>;
        compatible = "simple-bus";
        ranges:

        gpio@00000000 {
            #gpio-cells = <0x2>;
            #interrupt-cells = <0x2>;
            compatible = "generic-uio";
            gpio-controller;
            interrupt-controller;
            interrupt-parent = <0x0>;
            interrupts = <0x0 0x5d 0x4>; /* IRQ 125 */
            reg = <0x0 0xa0010000 0x0 0x10000>;
            xlnx,all-inputs = <0x0>;
            xlnx,all-inputs-2 = <0x0>;
            xlnx,all-outputs = <0x0>;
            xlnx,all-outputs-2 = <0x0>;
            xlnx,dout-default = <0x0>;
            xlnx,dout-default-2 = <0x0>;
            xlnx,gpio-width = <0x1>;
            xlnx,gpio2-width = <0x28>;
            xlnx,interrupt-present = <0x1>;
            xlnx,io-dual = <0x0>;
            xlnx,tri-default = <0xffffffff>;
            xlnx,tri-default-2 = <0xffffffff>;
        };

        uio0 {
            compatible = "generic-uio";
            status = "okay";
            interrupt-controller;
            interrupt-parent = <0x0>;
            interrupts = <0x0 0x59 0x1>; /* IRQ 121, edge rising */
        };

        uio1 {
            compatible = "generic-uio";
            status = "okay";
            interrupt-controller;
            interrupt-parent = <0x0>;
            interrupts = <0x0 0x5a 0x2>; /* IRQ 122, edge falling */
        };

        uio2 {
            compatible = "generic-uio";
            status = "okay";
            interrupt-controller;
            interrupt-parent = <0x0>;
            interrupts = <0x0 0x5b 0x4>; /* IRQ 123, level high */
        };

        uio3 {
            compatible = "generic-uio";
            status = "okay";
            interrupt-controller;
            interrupt-parent = <0x0>;
            interrupts = <0x0 0x5c 0x0>; /* IRQ 124, level low */
        };

        chosen {
            bootargs = "earlycon clk_ignore_unused uio_pdrv_genirq.of_id=generic-uio";
            stdout-path = "serial0:115200n8";
        };
    };
};

```

*\*The UIO\_PDRV\_GENIRQ driver never use fixed compatible table now. So there are two ways to make this driver to match the UIO device node in DTS.*

1. bootargs use "uio\_pdrv\_genirq.of\_id=generic-uio"
2. insmod uio\_pdrv\_genirq.ko of\_id=generic-uio when install the driver

## Test

Boot to kernel with uio\_pdrv\_genirq.ko auto loaded. The /dev/uio has been created.

```

root@zcu102-pl2ps_irq:~# lsmod
Not tainted
uio_pdrv_genirq 16384 0 - Live 0xffffffff8000940000
root@zcu102-pl2ps_irq:~# ls /dev/u
uio0      uio1      uio2      urandom

```

Check the /proc/interrupts.



```

root@zcui02-pl2ps_irq:~# cat /proc/interrupts
CPU0 CPU1 CPU2 CPU3
2: 0 0 0 0 GICv2 29 Level arch_timer
3: 1227 1086 2473 1336 GICv2 30 Level arch_timer
10: 0 0 0 0 GICv2 67 Level zynqmp_pm
13: 0 0 0 0 GICv2 156 Level zynqmp-dma
14: 0 0 0 0 GICv2 157 Level zynqmp-dma
15: 0 0 0 0 GICv2 158 Level zynqmp-dma
16: 0 0 0 0 GICv2 159 Level zynqmp-dma
17: 0 0 0 0 GICv2 160 Level zynqmp-dma
18: 0 0 0 0 GICv2 161 Level zynqmp-dma
19: 0 0 0 0 GICv2 162 Level zynqmp-dma
20: 0 0 0 0 GICv2 163 Level zynqmp-dma
22: 0 0 0 0 GICv2 109 Level zynqmp-dma
23: 0 0 0 0 GICv2 110 Level zynqmp-dma
24: 0 0 0 0 GICv2 111 Level zynqmp-dma
25: 0 0 0 0 GICv2 112 Level zynqmp-dma
26: 0 0 0 0 GICv2 113 Level zynqmp-dma
27: 0 0 0 0 GICv2 114 Level zynqmp-dma
28: 0 0 0 0 GICv2 115 Level zynqmp-dma
29: 0 0 0 0 GICv2 116 Level zynqmp-dma
33: 0 0 0 0 GICv2 49 Level cdns-l2c
34: 0 0 0 0 GICv2 50 Level cdns-l2c
35: 0 0 0 0 GICv2 42 Level ff960000.memory-controller
36: 0 0 0 0 GICv2 150 Level nwl_pcie:misc
41: 14 0 0 0 GICv2 47 Level ff070000.spi
42: 0 0 0 0 GICv2 58 Level ffa60000.rtc
43: 0 0 0 0 GICv2 59 Level ffa60000.rtc
44: 0 0 0 0 GICv2 165 Level ahci-ceva[f80c0000.ahci]
45: 170 0 0 0 GICv2 81 Level mmc0
46: 126 0 0 0 GICv2 53 Level xuartps
48: 0 0 0 0 GICv2 145 Edge fd4d0000.watchdog
49: 0 0 0 0 GICv2 88 Level am-sirq
50: 0 0 0 0 GICv2 151 Level fd4b0000.dp
51: 0 0 0 0 GICv2 154 Level fd4c0000.dma
52: 0 0 0 0 GICv2 125 Level gpio
53: 0 0 0 0 GICv2 121 Edge uio
55: 0 0 0 0 GICv2 123 Level uio
231: 0 0 0 0 GICv2 97 Level ahci-hcd:usb1
IP10: 1911 1830 803 1157 Rescheduling interrupts
IP11: 82 96 90 27 Function call interrupts
IP12: 0 0 0 0 CPU stop interrupts
IP13: 9 0 3 2 Timer broadcast interrupts
IP14: 0 0 0 0 IRQ work interrupts
IP15: 0 0 0 0 CPU wake-up interrupts
Err: 0

```

Why no IRQ 122-edge falling and 124-level low?

\*These two type interrupt is not support by hardware. See the kernel dmesg log below.

```

4.421366] udevd[1637]: starting udevd-3.2
4.457888] random: udevd: uninitialized urandom read (16 bytes read)
4.464422] random: udevd: uninitialized urandom read (16 bytes read)
4.476735] genirq: Setting trigger mode 2 for irq 54 failed (gic_set_type+0x0/0x48)
4.484964] uio_pdrv_genirq amba_pl0:uio01: unable to register uio device
4.485460] random: udevd: uninitialized urandom read (16 bytes read)
4.488256] random: udevd: uninitialized urandom read (16 bytes read)
4.504776] uio_pdrv_genirq: probe of amba_pl0:uio01 failed with error -22
4.512875] genirq: Setting trigger mode 8 for irq 56 failed (gic_set_type+0x0/0x48)
4.519904] uio_pdrv_genirq amba_pl0:uio03: unable to register uio device
4.520800] uio_pdrv_genirq: probe of amba_pl0:uio03 failed with error -22
4.908660] random: dd: uninitialized urandom read (512 bytes read)
5.050601] random: dropbearkey: uninitialized urandom read (32 bytes read)

```

Switch each DIP0(SW13.8) and DIP2(SW13.6) one time to trigger the interrupts.

```

root@zcui02-pl2ps_irq:~# cat /proc/interrupts
CPU0 CPU1 CPU2 CPU3
2: 0 0 0 0 GICv2 29 Level arch_timer
3: 1671 2059 2859 1069 GICv2 30 Level arch_timer
10: 0 0 0 0 GICv2 67 Level zynqmp_pm
13: 0 0 0 0 GICv2 156 Level zynqmp-dma
14: 0 0 0 0 GICv2 157 Level zynqmp-dma
15: 0 0 0 0 GICv2 158 Level zynqmp-dma
16: 0 0 0 0 GICv2 159 Level zynqmp-dma
17: 0 0 0 0 GICv2 160 Level zynqmp-dma
18: 0 0 0 0 GICv2 161 Level zynqmp-dma
19: 0 0 0 0 GICv2 162 Level zynqmp-dma
20: 0 0 0 0 GICv2 163 Level zynqmp-dma
22: 0 0 0 0 GICv2 109 Level zynqmp-dma
23: 0 0 0 0 GICv2 110 Level zynqmp-dma
24: 0 0 0 0 GICv2 111 Level zynqmp-dma
25: 0 0 0 0 GICv2 112 Level zynqmp-dma
26: 0 0 0 0 GICv2 113 Level zynqmp-dma
27: 0 0 0 0 GICv2 114 Level zynqmp-dma
28: 0 0 0 0 GICv2 115 Level zynqmp-dma
29: 0 0 0 0 GICv2 116 Level zynqmp-dma
33: 0 0 0 0 GICv2 49 Level cdns-l2c
34: 0 0 0 0 GICv2 50 Level cdns-l2c
35: 0 0 0 0 GICv2 42 Level ff960000.memory-controller
36: 0 0 0 0 GICv2 150 Level nwl_pcie:misc
41: 14 0 0 0 GICv2 47 Level ff070000.spi
42: 0 0 0 0 GICv2 58 Level ffa60000.rtc
43: 0 0 0 0 GICv2 59 Level ffa60000.rtc
44: 0 0 0 0 GICv2 165 Level ahci-ceva[f80c0000.ahci]
45: 170 0 0 0 GICv2 81 Level mmc0
46: 109 0 0 0 GICv2 53 Level xuartps
48: 0 0 0 0 GICv2 145 Edge fd4d0000.watchdog
49: 0 0 0 0 GICv2 88 Level am-sirq
50: 0 0 0 0 GICv2 151 Level fd4b0000.dp
51: 0 0 0 0 GICv2 154 Level fd4c0000.dma
52: 0 0 0 0 GICv2 125 Level gpio
53: 1 0 0 0 GICv2 121 Edge uio
55: 1 0 0 0 GICv2 123 Level uio
231: 0 0 0 0 GICv2 97 Level ahci-hcd:usb1
IP10: 1912 1858 818 1168 Rescheduling interrupts
IP11: 113 127 121 27 Function call interrupts
IP12: 0 0 0 0 CPU stop interrupts
IP13: 69 0 49 75 Timer broadcast interrupts
IP14: 0 0 0 0 IRQ work interrupts
IP15: 0 0 0 0 CPU wake-up interrupts
Err: 0

```

## Test the DIP UIO

Refer to <https://01.org/linuxgraphics/gfx-docs/drm/driver-api/uio-howto.html>

Using uio\_pdrv\_genirq for platform devices

Especially in embedded devices, you frequently find chips where the irq pin is tied to its own dedicated interrupt line. In such cases, where you can be really sure the interrupt is not shared, we can take the concept of `uio_pdrv_genirq` one step further and use a generic interrupt handler. That's what `uio_pdrv_genirq` does.

The setup for this driver is the same as described above for `uio_pdrv`, except that you do not implement an interrupt handler. The `handler` element of `struct uio_info` must remain `NULL`. The `irq_flags` element must not contain `IRQ_SHARED`.

You will set the `name` element of `struct platform_device` to "uio\_pdrv\_genirq" to use this driver.

The generic interrupt handler of `uio_pdrv_genirq` will simply disable the interrupt line using `disable_irq_wake()`. After doing its work, userspace can reenable the interrupt by writing `0x00000001` to the UIO device file. The driver already implements an `irq_control()` to make this possible, you must not implement your own.

Using `uio_pdrv_genirq` not only saves a few lines of interrupt handler code. You also do not need to know anything about the chip's internal registers to enable the kernel part of the driver. All you need to know is the irq number of the pin the chip is connected to.

So do the enable IRQ by “echo 0x1 > /dev/uioX” (write system call which trigger the irqcontrol) after each time the interrupt triggered by the DIP switch. The two DIP UIO is /dev/uio1 and /dev/uio2 in kernel.

```
root@rcul02-pl2ps_irq:/sys/class/uio/uio0# echo 0x1 > /dev/uio1
+ UIO irqcontrol: irq_on=17014192, priv->flags=0x00000001| 2782.948186| + UIO irqcontrol: priv->flags=0x00000000
root@rcul02-pl2ps_irq:/sys/class/uio/uio0# cat /proc/interrupts | grep uio
53:      0          0          0          0      GICv2 121 Edge      uio
55:      0          0          0          0      GICv2 123 Level    uio
root@rcul02-pl2ps_irq:/sys/class/uio/uio0# echo 0x1 > /dev/uio1
+ UIO irqcontrol: irq_on=17014192, priv->flags=0x00000001| 2822.236270| + UIO irqcontrol: priv->flags=0x00000001
root@rcul02-pl2ps_irq:/sys/class/uio/uio0# echo 0x1 > /dev/uio2
+ UIO irqcontrol: irq_on=17014192, priv->flags=0x00000001| 2822.744182| + UIO irqcontrol: priv->flags=0x00000000
root@rcul02-pl2ps_irq:/sys/class/uio/uio0# cat /proc/interrupts | grep uio
53:      0          0          0          0      GICv2 121 Edge      uio
55:      1          0          0          0      GICv2 123 Level    uio
root@rcul02-pl2ps_irq:/sys/class/uio/uio0# echo 0x1 > /dev/uio1
+ UIO irqcontrol: irq_on=17014192, priv->flags=0x00000001| 3021.808260| + UIO irqcontrol: priv->flags=0x00000000
root@rcul02-pl2ps_irq:/sys/class/uio/uio0# echo 0x1 > /dev/uio2
+ UIO irqcontrol: irq_on=17014192, priv->flags=0x00000001| 3022.548198| + UIO irqcontrol: priv->flags=0x00000000
root@rcul02-pl2ps_irq:/sys/class/uio/uio0# cat /proc/interrupts | grep uio
53:      0          0          0          0      GICv2 121 Edge      uio
55:      2          0          0          0      GICv2 123 Level    uio
root@rcul02-pl2ps_irq:/sys/class/uio/uio0#
```

Another test with user application.



pin-uio-test.c    pin-uio-test

```
root@rcul02-pl2ps_irq:/media# ./pin-uio-test -d /dev/uio1
pin UIO test
started uio test driver.
[ 50.206477] + UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 50.217095| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 1
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 68.041858| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 2
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 69.402641| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 3
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 70.578310| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 4
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 70.592732| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 5
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 72.537413| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 6
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 72.549758| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 7
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 72.399669| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 8
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 73.411025| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 9
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 74.245022| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 10
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 76.349610| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 11
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 78.032263| + UIO irqcontrol: priv->flags=0x00000000
Interrupts: 12
+ UIO irqcontrol: irq_on=1, priv->flags=0x00000001| 78.991805| + UIO irqcontrol: priv->flags=0x00000000
```

Run the application and trigger the interrupt by DIP switch.

## Test the GPIO UIO which is /dev/uio0 in kernel



gpio-uio-test.c    gpio-uio-test

This test application mmap out the registers from hardware to user space. Then enable all the IRQ bits in GIER and IP\_IER registers and dump out all the registers' values. Please refer to pg144-axi-gpio.pdf for the IP.



Table 2-4 shows the AXI GPIO registers and their addresses.

Table 2-4: Registers

Address Space Offset <sup>(1)</sup>	Register Name	Access Type	Default Value	Description
0x0000	GPIO_DATA	R/W	0x0	Channel 1 AXI GPIO Data Register.
0x0004	GPIO_TRI	R/W	0x0	Channel 1 AXI GPIO 3-state Control Register.
0x0008	GPIO2_DATA	R/W	0x0	Channel 2 AXI GPIO Data Register.
0x000C	GPIO2_TRI	R/W	0x0	Channel 2 AXI GPIO 3-state Control Register.
0x011C	GIER <sup>(1)</sup>	R/W	0x0	Global Interrupt Enable Register.
0x0128	IP IER <sup>(2)</sup>	R/W	0x0	IP Interrupt Enable Register (IP IER).
0x0120	IP ISR <sup>(3)</sup>	R/TOW <sup>(2)</sup>	0x0	IP Interrupt Status Register.

**Notes:**

1. Interrupt registers are available only if AXI GPIO is compiled using the **Enable Interrupt** parameter.
2. Toggle-On-Write (TOW) access toggles the status of the bit when a value of 1 is written to the corresponding bit.
3. Address Space Offset is relative to C\_BASEADDR assignment.

Test step and log.

```

root@zcu102-pl2ps_irq:/media# cat /proc/interrupts | grep gpio
52:          0          0          0          0      GICv2 125 Level      gpio
root@zcu102-pl2ps_irq:/media# ./gpio-uio-test -d /dev/uio0 -o 1
GPIO UIO test.
./gpio-uio-test: GIER: 00000000
./gpio-uio-test: [ 63.636574] + UIO irqcontrol: irq_on=1, priv->flags=0x00000000IP_IER: 00000000
./gpio-uio-test: IP_ISR: 00000000
./gpio-uio-test: Enable All Interrupts in Regs
[ 63.647880] + UIO irqcontrol: priv->flags=0x00000000
./gpio-uio-test: GIER: 80000000
./gpio-uio-test: IP_IER: 00000001
./gpio-uio-test: IP_ISR: 00000001
root@zcu102-pl2ps_irq:/media# cat /proc/interrupts | grep gpio
52:          2          0          0          0      GICv2 125 Level      gpio
root@zcu102-pl2ps_irq:/media# cat /proc/interrupts | grep gpio
52:          2          0          0          0      GICv2 125 Level      gpio
root@zcu102-pl2ps_irq:/media# ./gpio-uio-test -d /dev/uio0 -o 1
GPIO UIO test.
./gpio-uio-test: GIER: 80000000
./gpio-uio-test: + UIO irqcontrol: irq_on=1, priv->flags=0x00000000IP_IER: 00000001
./gpio-uio-test: IP_ISR: 00000001
./gpio-uio-test: Enable All Interrupts in Regs
[ 69.482470] + UIO irqcontrol: priv->flags=0x00000000
./gpio-uio-test: GIER: 80000000
./gpio-uio-test: IP_IER: 00000001
./gpio-uio-test: IP_ISR: 00000000
root@zcu102-pl2ps_irq:/media# cat /proc/interrupts | grep gpio
52:          2          0          0          0      GICv2 125 Level      gpio
root@zcu102-pl2ps_irq:/media# ./gpio-uio-test -d /dev/uio0 -o 1
GPIO UIO test.
./gpio-uio-test: GIER: 80000000
./gpio-uio-test: + UIO irqcontrol: irq_on=1, priv->flags=0x00000000IP_IER: 00000001
./gpio-uio-test: IP_ISR: 00000000
./gpio-uio-test: Enable All Interrupts in Regs
[ 95.678488] + UIO irqcontrol: priv->flags=0x00000000
./gpio-uio-test: GIER: 80000000
./gpio-uio-test: IP_IER: 00000001
./gpio-uio-test: IP_ISR: 00000001
root@zcu102-pl2ps_irq:/media# cat /proc/interrupts | grep gpio
52:          4          0          0          0      GICv2 125 Level      gpio
root@zcu102-pl2ps_irq:/media# █

```

## Reference

<https://01.org/linuxgraphics/gfx-docs/drm/driver-api/uio-howto.html>

[Xilinx]

[Simple PL-PS interrupt difficulty \(Vivado 2017.2 + Petalinux\)](#)

[PL to PS interrupt in linux /proc/interrupts](#)

[UIO Interrupts on Zynq](#) (refer to <https://embeddedcentric.com/interrupts/> )

[UIO interrupt with PS GPIO](#)

[Petalinux 2016.3 UIO](#)

[petalinux not creating uio](#)

<http://www.wiki.xilinx.com/GIC>

---

[ulmage.FIT]

[https://github.com/wowotechX/u-boot/tree/x\\_integration/doc/ulmage.FIT](https://github.com/wowotechX/u-boot/tree/x_integration/doc/ulmage.FIT)

<http://www.wiki.xilinx.com/U-Boot+Images>

**UG1085** - Zynq UltraScale+ MPSoC Technical Reference Manual

**UG1182** - ZCU102 Board User Guide

**PG144** - axi-gpio

## Questions: