How to use GPIO signals

From RidgeRun Developer Connection

Contents

GPIO Usage from a Linux Application

Overview GPIO Signals

The following table summarizes the steps to configuring and using GPIO signals from a Linux application.

Step Number	Action	Description
1	Configure the kernel for GPIO support in sysfs	Allow GPIO configuration and control from Linux applications (user space). The GPIO shoulw up in the *system file system*, sysfs, at `/sys/class/gpio`
2	Export GPIO to user space	Each GPIO is are not accessible from user space until the GPIO has been exported. You can only export a GPIO that isn't *owned* by a Linux kernel driver
3	Configure GPIO for input or output	To avoid hardware issues where two devices are driving the same signal, GPIOs default to be configured as an input. If you want to use the GPIO as an output, you need to change the configuration
4	Configure GPIO an an interrupt source	If you have a GPIO that is an input, and you have an application you want to block waiting for the GPIO to change level, you can configure the GPIO as an interrupt source. You also need to configure if the interrupt occurs when the GPIO signal has a rising edge, a falling edge, or interrupts on both rising and falling edges. Once configured as an interrupt, your application can read the value file and the read will block until the

		interrupt occurs, then your application will return from the read system call and continue running.

The sysfs directory `/sys/class/gpio` contains subdirectories and files that are used for configuring and using GPIO signals from a Linux application.

File or directory	Meaning	Notes
/sys/class/gpio	sysfs GPIO subdirectory	Linux applications can configuration and use GPIO signals by accessing files in this subdirectory.
/sys/class/gpio/export	Write-only file to expose a GPIO	Before a Linux application can configuration and use a GPIO, the GPIO first has to be *exported to user space* by writing the GPIO number to this file.
/sys/class/gpio/gpio<<<number>>>	Subdirectory for configuring and reading a specific GPIO signal	Once a GPIO has been exported to user space, a new directory appears with a set of files that allow the GPIO to be configured and used by a Linux application
/sys/class/gpio/gpio<<<number>>>/direction	Read-write supporting values of *in* and *out*	
/sys/class/gpio/gpio<<<number>>>/value	Read-write supporting values of *0* and *1*	
/sys/class/gpio/gpio<<<number>>>/edge	Read-write supporting values of *rising*, *falling*, and *both*	Check your processor documentation. Not all GPIO pins support interrupts.

Configure the kernel for GPIO support in sysfs

```
Symbol: GPIO_SYSFS [=y]
  Prompt: /sys/class/gpio/... (sysfs interface)
    Defined at drivers/gpio/Kconfig:51
    Depends on: GPIOLIB && SYSFS && EXPERIMENTAL
     Location:
      -> Kernel configuration
        -> Device Drivers
         -> GPIO Support (GPIOLIB [=y])
```

Enable GPIO access from user space

```
GPIO=22

cd /sys/class/gpio
```

```
ls
echo 22 > export
ls
```


Notice on the first *ls* that *gpio22* doesn't exist, but does after you export GPIO 22 to user space.

```
cd /sys/class/gpio/gpio$GPIO
ls
```

There are files to set the direction and retrieve the current value.

```
echo "in" > direction
cat value
```

You can configure the GPIO for output and set the value as well.

```
echo "out" > direction
echo 1 > value
```

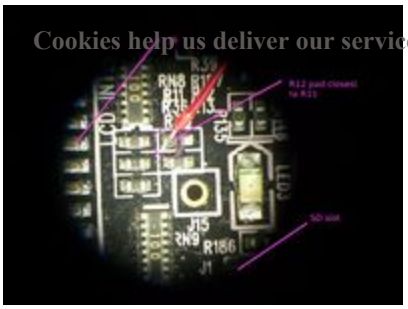
GPIO interrupts from user space

References

- "How to implement an interrupt driven GPIO input in Linux" Posted by Cliff Brake BEC Systems on 2009-01-10. URL: http://bec-systems.com/site/281/how-to-implement-an-interrupt-driven-gpio-input-in-linux (accessed on 16 June 2018)
- "devfs Gpio and interrupt" Content of an email from linux-newbie thread of spinics.net website. URL: https://www.spinics.net/lists/linux-newbie/msg01028.html (accessed on 16 June 2018)
- "GPIOs and Linux - Communicating with the outside world" Presentation material from elinux.org. URL: https://elinux.org/images/d/d4/Celf-gpio.odp (accessed on 16 June 2018)
- "GPIO sysfs Interface" Document from Analog Devices Open Source Projects For Blackfin Processors. URL: https://docs.blackfin.uclinux.org/doku.php?id=linux-kernel:drivers:gpio-sysfs (accessed on 16 June 2018)

LeopoardBoard 365 GPIO 0 connection

On the LeopardBoard 365, the only GPIO I could find that was usable for interrupt input is GPIO0, also called CMOS_TRIGGER in the schematics. In looking at the schematics resistor R12 is not loaded and one of the pads connects to CMOS_TRIGGER. This R12 pad is the one closest to R11. If you hold the leopardboard 365 with the SD card slot facing you and rotate the board until the SD card slot is on the bottom edge, the the R12 pads are to the right of J6 and to the left of the SD card slot upper left corner.



Using poll() to monitor for GPIO 0 change

The gpio-int-test.c program (or gpiopin.cpp for those who prefer C++) shows one way of using the sysfs file /sys/class/gpio/gpio0/value to block program execution using poll() until the input level on GPIO0 changes. The tricky part was figuring out to use POLLPRI instead of POLLIN as the event to monitor. You must have GPIO support in sysfs for this program to work (or you will not see the /sys/class/gpio directory).

The gpio-int-test.c program uses poll() to wake up every 3 seconds (using poll() timeout mechanism) at which time it prints a period. The poll() function is also watching for input from stdin and for an interrupt from GPIO 0.

Here is an example output. I started gpio-int to watch GPIO 0. I waited around 12 seconds (4 timeout periods), then pressed the letter 'a' twice followed by enter key. Then I shorted the haywire to 3.3V that is accessible on pin 5 on the JTAG connector. JTAG pin 5 is across from the JTAG missing pin). I exited the program using cntl-C.

```
/root # gpio-int 0

....aa

poll() stdin read 0xA61

poll() stdin read 0xA61

poll() stdin read 0xA0A
..
poll() GPIO 0 interrupt occurred (len 0)

poll() GPIO 0 interrupt occurred (len 0)

poll() GPIO 0 interrupt occurred (len 0)

poll() GPIO 0 interrupt occurred (len 0)
..^C
```

Viewing GPIO Configuration

You can use debugfs (http://lwn.net/Articles/334546/) to videw the current GPIO configuration. You may also be able to use debugfs to see if the GPIO pin is multiplex as a GPIO or is dedicated to some other function.

Configure the kernel to enable debugfs:

```
Symbol: DEBUG_FS [=y]
   Prompt: Debug Filesystem
     Defined at lib/Kconfig.debug:77
```

```
    Depends on: SYSFS
    Location:
      -> Kernel configuration
        -> Kernel hacking
```


Boot the target hardware and mount debugfs:

```
mount -t debugfs none /sys/kernel/debug
```

Dump the GPIO configuration.

```
cat /sys/kernel/debug/gpio
```

Dump the pin multiplexing configuration.

```
cat /sys/kernel/debug/omap_mux/board      # for OMAP
cat /sys/kernel/debug/dm365_mux           # for DM36x
```

Example shell script making it easy to set GPIOs from the command line

If you want to have a simple way to control a GPIO signal from the Linux command line, try the **gpio.sh** script below.

For examaple, if you want to read the value of GPIO 72 without setting its direction, try

```
gpio.sh 72
```

If you want to force GPIO 35 to be in input and read the current value, try

```
gpio.sh 35 in
```

If you want to configure GPIO 4 to be an output and set the value high, try

```
gpio.sh 4 out 1
```

For the script below to work, you need to first make sure you have build busybox with printf enabled.

```
#!bin/sh

show_usage()
{
    printf "\ngpio.sh <gpio pin number> [in|out [<value>]]\n"
}

if [ \( $# -eq 0 \) -o \( $# -gt 3 \) ] ; then
    show_usage
    printf "\n\nERROR: incorrect number of parameters\n"
    exit 255
fi
```

```
#doesn't hurt to export a gpio more than once
echo $1 > /sys/class/gpio/export

if [  $# -eq 1 ] ; then
   cat /sys/class/gpio/gpio$1/value
   exit 0
fi

if [ \( "$2" != "in" \) -a  \( "$2" != "out" \) ] ; then
    show_usage
    printf "\n\nERROR: second parameter must be 'in' or 'out'\n"
    exit 255
fi

echo $2 > /sys/class/gpio/gpio$1/direction

if [  $# -eq 2 ] ; then
   cat /sys/class/gpio/gpio$1/value
   exit 0
fi


VAL=$3

if [ $VAL -ne 0 ] ; then
    VAL=1
fi

echo $VAL > /sys/class/gpio/gpio$1/value
```


Retrieved from "http://developer.ridgerun.com/wiki/index.php?title=How_to_use_GPIO_signals&oldid=17054"

Categories: HowTo | Whitepaper

- This page was last edited on 13 July 2018, at 03:20.