How to control a GPIO in userspace

Contents

Stable: 11.06.2020 - 08:19 / Revision: 11.06.2020 - 08:31

A quality version of this page, approved on *11 June 2020*, was based off this revision.

Contents

1 Purpose

This article shows two ways to control a GPIO in userspace:

- using libgpiod
- by writing an application

2 GPIO control through libgpiod

libgpiod provides a C library and tools for interacting with the linux GPIO character device (gpiod stands for GPIO device). See the libgpiod repository[1] for further explanation.

- **gpiodetect**
  - List all gpiochips present on the system
  - Usage:

```
Board $> gpiodetect
gpiochip11 [GPIOZ] (16 lines)
...
gpiochip0 [GPIOA] (16 lines)
```

- **gpioinfo**
  - list all lines of specified gpiochips, their names, consumers, and their settings
  - Usage:

```
Board $>gpioinfo
gpiochip11 - 16 lines:
       line   0:      unnamed       unused   input  active-high
       line   1:      unnamed       unused   input  active-high
       ...
```

or

```
                                Comments

Board $>gpioinfo gpiochip0    -->to only print gpiochip0 lines
```

- **gpioget**
  - Read the values of the specified GPIO lines (not valid if the line is already requested). The line will be then configured as input.

```
                                Comments

Board $>gpioget gpiochip0 5   -->to get value of GPIO PA5
0                             -->means the line is driven low
```

- **gpioset**
  - Set the values of the specified GPIO lines, potentially keeping the lines exported, and wait until timeout, user input or signal (not valid if the line is already requested).

```
Board $>gpioset gpiochip0 14=0  -->to set GPIO PA14 low // green led on
Board $>gpioset gpiochip0 14=1  -->to set GPIO PA14 high // green led off
```

3 GPIO control through your own application

3.1 Purpose

This application toggles GPIO PA14 (GPIO bank A, line 14). On STM32MP15_Evaluation_boards or STM32MP15_Discovery_kits GPIO PA14 is connected to the green LED.

This application must be cross compiled with same toolchain as the Kernel.

3.2 Code

```
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <unistd.h>

#include <linux/gpio.h>

int main(int argc, char **argv)
{
        struct gpiohandle_request req;
        struct gpiohandle_data data;
        char chrdev_name[20];
        int fd, ret;

        strcpy(chrdev_name, "/dev/gpiochip0");

        /*  Open device: gpiochip0 for GPIO bank A */
        fd = open(chrdev_name, 0);
        if (fd == -1) {
                ret = -errno;
                fprintf(stderr, "Failed to open %s\n", chrdev_name);

                return ret;
        }

        /* request GPIO line: GPIO_A_14 */
        req.lineoffsets[0] = 14;
        req.flags = GPIOHANDLE_REQUEST_OUTPUT;
        memcpy(req.default_values, &data, sizeof(req.default_values));
        strcpy(req.consumer_label, "led_gpio_a_14");
        req.lines  = 1;

        ret = ioctl(fd, GPIO_GET_LINEHANDLE_IOCTL, &req);
        if (ret == -1) {
                ret = -errno;
                fprintf(stderr, "Failed to issue GET LINEHANDLE IOCTL (%d)\n",
                        ret);
        }
        if (close(fd) == -1)
                perror("Failed to close GPIO character device file");

        /*  Start led blinking */
        while(1) {

                data.values[0] = !data.values[0];
```

```
                ret = ioctl(req.fd, GPIOHANDLE_SET_LINE_VALUES_IOCTL, &data);
                if (ret == -1) {
                        ret = -errno;
                        fprintf(stderr, "Failed to issue %s (%d)\n",
                                        "GPIOHANDLE_SET_LINE_VALUES_IOCTL", ret);
                }
                sleep(1);
        }

        /*  release line */
        ret = close(req.fd);
        if (ret == -1) {
                perror("Failed to close GPIO LINEHANDLE device file");
                ret = -errno;
        }
        return ret;
}
```

3.3 Build application

See Adding_Linux_user_space_applications to build this application.

4 References

- libgpiod repository

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Light-emitting diode