libgpiod

libgpiod - C library and tools for interacting with the linux GPIO character device

Since linux 4.8 the GPIO sysfs interface is deprecated. User space should use the character device instead. This library encapsulates the ioctl calls and data structures behind a straightforward API.

RATIONALE

The new character device interface guarantees all allocated resources are freed after closing the device file descriptor and adds several new features that are not present in the obsolete sysfs interface (like event polling, setting/reading multiple values at once or open-source and open-drain GPIOs).

Unfortunately interacting with the linux device file can no longer be done using only standard command-line tools. This is the reason for creating a library encapsulating the cumbersome, ioctl-based kernel-userspace interaction in a set of convenient functions and opaque data structures.

Additionally this project contains a set of command-line tools that should allow an easy conversion of user scripts to using the character device.

BUILDING

This is a pretty standard autotools project. It does not depend on any libraries other than the standard C library with GNU extensions.

The autoconf version needed to compile the project is 2.61.

Recent (as in >= v4.8) kernel headers are also required for the GPIO user API definitions.

To build the project (including command-line utilities) run:

```
./autogen.sh
./configure --enable-tools=yes --prefix=<install path>
make
make install
```

TOOLS

There are currently six command-line tools available:

- gpiodetect - list all gpiochips present on the system, their names, labels and number of GPIO lines
- gpioinfo - list all lines of specified gpiochips, their names, consumers, direction, active state and additional flags

- gpioget - read values of specified GPIO lines
- gpioset - set values of specified GPIO lines, potentially keep the lines exported and wait until timeout, user input or signal
- gpiofind - find the gpiochip name and line offset given the line name
- gpiomon - wait for events on a GPIO line, specify which events to watch, how many events to process before exiting or if the events should be reported to the console

Examples:

```
# Read the value of a single GPIO line.
# gpioget gpiochip1 23
0

# Read two values at the same time. Set the active state of the lines
# to low.
# gpioget --active-low gpiochip1 23 24
1 1

# Set values of two lines, then daemonize and wait for a signal (SIGINT or
# SIGTERM) before releasing them.
# gpioset --mode=signal --background gpiochip1 23=1 24=0

# Set the value of a single line, then exit immediately. This is useful
# for floating pins.
# gpioset gpiochip1 23=1

# Find a GPIO line by name.
# gpiofind "USR-LED-2"
gpiochip1 23

# Toggle a GPIO by name, then wait for the user to press ENTER.
# gpioset --mode=wait `gpiofind "USR-LED-2"`=1

# Wait for three rising edge events on a single GPIO line, then exit.
# gpiomon --num-events=3 --rising-edge gpiochip2 3
event:  RISING EDGE offset: 3 timestamp: [    1151.814356387]
event:  RISING EDGE offset: 3 timestamp: [    1151.815449803]
event:  RISING EDGE offset: 3 timestamp: [    1152.091556803]

# Pause execution until a single event of any type occurs. Don't print
# anything. Find the line by name.
# gpiomon --num-events=1 --silent `gpiofind "USR-IN"`
```

TESTING

A comprehensive unit testing framework is included with the library and can be used to test both the

library code and the linux kernel user-space interface.

The minimum kernel version required to run the tests is 4.11. The tests work together with the gpio-mockup kernel module which must be enabled. NOTE: the module must not be built-in.

To build the testing executable run:

```
./configure --enable-tests
make check
```

As opposed to standard autotools projects, libgpiod doesn't execute any tests when invoking 'make check'. Instead the user must run them manually with superuser privileges:

```
sudo ./tests/unit/gpiod-unit
```

CONTRIBUTING

Contributions are welcome - please use github pull-requests and issues and stick to the linux kernel coding style when submitting new code.