GPIO driver for the REXYGEN system (the GPIODrv module)

User guide

REX Controls s.r.o.

Version 3.0.5
2025-07-04
Plzeň (Pilsen), Czech Republic

Contents

Chapter 1

The GPIODrv driver and the REXYGEN system

1.1 Introduction

This guide describes the use of the GPIODrv driver for accessing the GPIO pins of a generic device within the REXYGENcontrol system. The driver supports both input and output modes for the pins and is compatible with devices running Linux kernel 5.10 or higher. The driver was developed by REX Controls.

If you are new to REXYGEN, we recommend starting with the tutorials *Getting Started and Tutorials*, which:

- Introduce you to the basic principles of the REXYGEN system and its development tools [1],
- Help you with the installation and configuration of the REXYGEN system on your target device [2],
- Show you how to create a simple project including a basic user interface [3],
- Demonstrate how to configure the inputs and outputs of your device [4],
- Enable you to create your own graphical user interface [5].

1.2 Installation

To be able to configure the GPIODrv driver in the REXYGEN Studio development environment, it is necessary to have the **Linux General Purpose I/O driver** option checked in the **Select Components** step during the REXYGEN Studio installation. The easiest is to select the **Full installation** option.

The GPIODrv driver is part of the standard RexCore installation on the target device and should be installed regardless of the installation method described in [2].

If for some reason you need to install `GPIODrv` separately, you can do so from the device command line using the command:
`sudo apt-get install rex-gpiodrvt`

1.2.1 Enabling PWM in the Operating System

To use GPIO in PWM mode, it is necessary to activate this mode in the system. This can be achieved by adding the `pwm-2chan` dtoverlay to the system configuration file. For Raspberry Pi 5, for example, add the following line to the `/boot/firmware/config.txt` file:

```
dtoverlay=pwm-2chan
```

After modifying the configuration file, the device must be restarted with the following command:

```
sudo reboot
```

This configuration enables two PWM channels that can be controlled via GPIO pins.

Chapter 2

Including the driver in the project

The driver is included in the project as soon as the driver is added to the project main file and the inputs and outputs are connected in the control algorithms.

2.1 Adding the `GPIODrv` driver

The project main file with the `GPIODrv` driver included is shown in Figure 2.1.

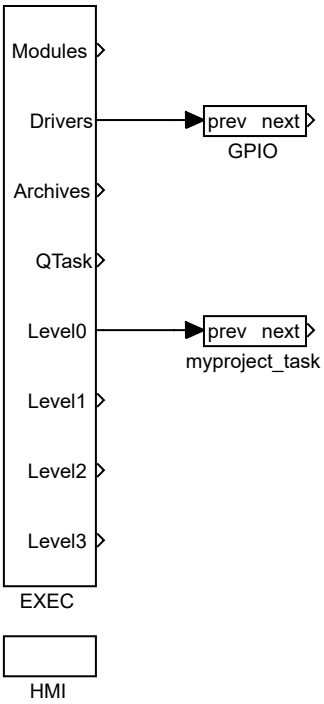

Figure 2.1: An example of including the `GPIODrv` driver in the main project file

The block of type `IODRV` connected to the `Drivers` output of the main `EXEC` block is used

to include the driver in the project. After opening `Block properties` (by double-clicking on the block), you can edit the parameter settings. The four most important parameters of the `IODRV` block are:

- `module` – name of the module linked to the driver, in this case `GPIODrv`
- `classname` – class of the driver
  - `GpioDrv` – A class for standard usage of GPIO. It allows configuring pins as standard digital inputs or outputs and managing their states in real time.
  - `PwmDrv` – A class for controlling GPIO in PWM (Pulse Width Modulation) mode. This mode is suitable for applications where signal intensity needs to be controlled, such as LED dimming or motor speed control.
  - `GpioSysDrv` – **Do not use!** - This option is retained for older versions of the Linux kernel.
- `cfgname` – name of the driver configuration file, but this driver does not use any so leave it blank
- `factor` – multiple of the EXEC block's tick parameter defining the driver's task execution period.

The `Configure` button opens the configuration dialog of the `GPIODrv` driver, which is described in Chapter 3.

All input and output signals of this driver begin with name of the `IODRV` block (see Fig. 2.1). It is recommended to rename the `IODRV` block according to the used class, `GPIO` or `PWM`.

2.2 Connecting the inputs and outputs in the control algorithm

The inputs and outputs of the drivers can be connected to the algorithm in individual tasks using several function blocks:

- To read a single value, it is convenient to use the `From` block.
- The `Goto` block is used to write a single value.
- Since the driver allows obtaining several inputs or setting several outputs under one symbolic name, it is advantageous to use blocks of four-fold, eight-fold and sixteen-fold inputs and outputs (`INQUAD`, `OUTQUAD`, `INOCT`, `OUTOCT` and `INHEXD`, `OUTHEXD`). The advantage of such use is an increase in the speed and partly also the clarity of the algorithms.

The affiliation of the From and Goto blocks to a given driver is given by their parameter Goto tag, which starts with the name of the IODRV block (see Fig. 2.1), continues with the separator __ (followed by two characters '_') and ends with the signal name. For blocks INQUAD, OUTQUAD, INOCT, OUTOCT, INHEXD and OUTHEXD, this identification string is entered directly into their name. A detailed description of the blocks can be found in the [6] manual.

The tasks are then connected in the main project file to the QTask, Level0, Level1, Level2 and Level3 outputs of the EXEC block. The QTask and Level0 outputs are reserved for the highest priority tasks, the Level3 output for the lowest priority tasks.

Below are examples for each platform supported by the GPIODrv driver. The examples use the From function block for reading, and the Goto function block for signal writing. Both of these blocks have a Goto tag parameter that is set to the name of the signal.

2.2.1 Standard usage of GPIO example

Due to the variability of GPIO compared to classic inputs and outputs, the above-described method of setting Goto tag is supplemented with an optional letter at the end. Depending on the letter, the following properties are set:

- U – activation of the pull-up resistor at the input,
- D – activation of the pull-down resistor at the input or setting the output to open drain mode,
- I – inverting the value at the input or output,
- S – setting the output to open source mode,
- H – setting the initial state of the output to logical 1 (on),
- L – setting the initial state of the output to logical 0 (off),
- B – setting the deBounce time for the input.

These properties can be combined. For example, the flag GPIO__0UI enables the pull-up resistor and simultaneously inverts the value on the input.

An example of use can be seen in Fig. 2.3. One From block used to connect a single input has its Goto tag parameter set to GPIO__23UI, while another block has it set to GPIO__24UI. The name following the underscores in the tag corresponds to the designated GPIO (pin designating depends on the specific device). The Goto block, used to connect a single output, has its Goto tag parameter set to GPIO__25.

Similarly, we can use, for example, the following flags:

- Goto, GPIO__22 – digital output 22
- Goto, GPIO__23H – digital output 23, which is set to logical 1 (on) immediately upon initialization

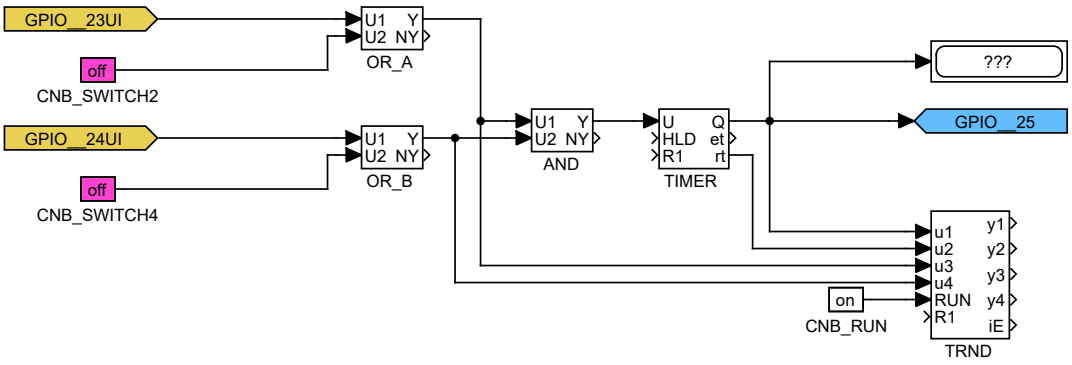

Figure 2.2: Example of GPIO connection in the control algorithm

- `From`, `GPIO__7U` – digital input 7 with internal pull-up resistor
- `From`, `GPIO__8D` – digital input 8 with internal pull-down resistor
- `From`, `GPIO__21` – digital input 21 without any internal pull-up/down resistor

The installation of the REXYGEN system includes a library of examples, where among other things, the section `0410_GPIO` is dedicated to the use of `GPIODrv`. The example `0410-00_IO_Flags` contains a library of usable inputs and outputs.

2.2.2 Example of using GPIO for PWM

For the `PwmDrv` class, only output flags are relevant. By default (without using any suffix in the `Goto tag`), the value written to the flag represents the desired duty cycle as a real number in the range <0;1>. Similarly to general GPIO usage, the `Goto tag` for PWM configuration can be extended with optional suffixes represented by individual letters. These suffixes allow the definition of the following properties:

- `I` – inverts the duty cycle value,
- `D` – the written value does not represent the duty cycle but the pulse length in nanoseconds as an integer,
- `P` – the period length in nanoseconds, which must be in the range <0;0xFFFF>.

These properties can be combined. For example, the designation `PWM__0ID` inverts the duty cycle value and sets the pulse length in nanoseconds.

The PWM channel number depends on the specific target device. For Raspberry Pi 1-4, GPIO 18 is assigned to channel `PWM__0` and GPIO 19 to channel `PWM__1`. On Raspberry Pi 5, GPIO 18 is available by default on channel `PWM__2` and GPIO 19 on channel `PWM__3`.

An example of usage on Raspberry Pi 5 is shown in Fig. 2.3. A `Goto` block used

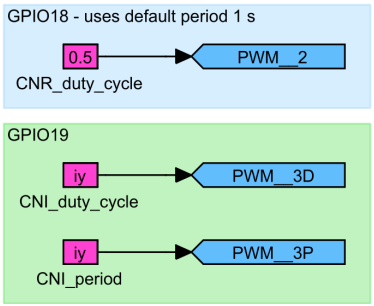

Figure 2.3: Example of PWM connection for Raspberry Pi 5 in a control algorithm

to connect a single output has the `Goto tag` parameter set to `PWM__2` for Raspberry Pi 5, which writes the desired duty cycle value to GPIO 18. Another `Goto` block with the `Goto tag` parameter set to `PWM__3D` writes the desired pulse length [ns] to GPIO 19 as a integer number. Additionally, the period length in nanoseconds must be set for GPIO 19 using a final `Goto` block with the `Goto tag` parameter set to `PWM__3P`.

The REXYGEN system installation includes a library of examples, including the `0411_PWM` section dedicated to using the `GPIODrv` driver. The `0411-00_IO_Flags` example contains a library of usable GPIO configurations.

Chapter 3

Configuration of the GPIODrv driver

The configuration dialog can be accessed by clicking the **Configure** button in the parameter dialog of the **IODRV** block (see Chapter 2.1).

3.1 Configuration for GpioDrv

To configure the **GpioDrv** class, the **GPIO Chip** parameter must be set to the name of the GPIO chip to be controlled. For example, use **gpiochip4** for Raspberry Pi 5. The appropriate value for the target device can be found in the terminal by executing the following command:

```
gpioinfo
```

3.2 Configuration for PwmDrv

To configure the **PwmDrv** class, the following parameters must be set:

- **PWM Chip** – the name of the PWM chip to be controlled. For example, **pwmchip2** for Raspberry Pi 5. The appropriate value for the target device can be found in the terminal by executing the following command:

  ```
  ls /sys/class/pwm
  ```

- **Default period [ns]** – the default PWM signal period length in nanoseconds.
- **Default duty cycle [ns]** – the default pulse length of the PWM signal in nanoseconds.

Important: Before using the **PwmDrv** class, ensure that the PWM interface has been activated in the system by modifying the configuration file and restarting the device (see Chapter 1.2).

Chapter 4

Troubleshooting

First and foremost, it is advisable to explore the example libraries related to using `GPIODrv` with the specific device.

If unexpected or incorrect input or output values appear in the diagnostic tools of the REXYGEN system, it is recommended to verify their functionality independently of the REXYGEN system. Furthermore, the configuration should be carefully checked. The most common errors include:

Configuration error – incorrect selection of the chip, pin, or pin mode.

Hardware error – incorrect wiring.

Pin access error – the pin is being used by another device (SPI bus, I2C bus, serial port) or by another program.

In the case that the given input or output works with other software tools and does not work in the REXYGEN system, report the problem to us, please. E-mail is preferred, reach us at `support@rexygen.com`. Please include the following information in your description to help us process your request as soon as possible:

- Identification of the REXYGEN system you are using. Simply export it to a file using the REXYGEN Studio program (Target → Licensing → Export).
- Short and accurate description of your problem.
- The configuration files of the REXYGEN system (`.mdl` files) reduced to the simplest case which still demonstrates the problematic behavior.

Bibliography

[1] REX Controls s.r.o.. *Getting started with REXYGEN*, 2024. →.

[2] REX Controls s.r.o.. *RexCore (REXYGEN runtime core) Installation*, 2024. →.

[3] REX Controls s.r.o.. *First Project*, 2024. →.

[4] REX Controls s.r.o.. *I/O Configuration of Target Platforms*, 2024. →.

[5] REX Controls s.r.o.. *HMI creation in REXYGEN HMI Designer*, 2024. →.

[6] REX Controls s.r.o.. *Function blocks of REXYGEN – reference manual*, 2024. →.

Documentation reference number: 17331