
AT07903: SAM4L General-Purpose Input/Output (GPIO) Driver

APPLICATION NOTE

Introduction

This driver for Atmel® | SMART ARM®-based microcontrollers provides an interface for the configuration and management of the on-chip General-Purpose Input/Output (GPIO) controller with both interrupt and event support.

Devices from the following series can use this module:

- Atmel | SMART SAM4L

The outline of this documentation is as follows:

- [Prerequisites](#)
- [Module Overview](#)
- [Special Considerations](#)
- [Extra Information](#)
- [Examples](#)
- [API Overview](#)

Table of Contents

Introduction.....	1
1. Software License.....	3
2. Prerequisites.....	4
3. Module Overview.....	5
4. Special Considerations.....	6
5. Extra Information.....	7
6. Examples.....	8
7. API Overview.....	9
7.1. Variable and Type Definitions.....	9
7.1.1. Type gpio_pin_callback_t.....	9
7.2. Function Definitions.....	9
7.2.1. Interrupt Support.....	9
7.2.2. Peripheral Event System Support.....	10
8. Extra Information for GPIO.....	12
8.1. Acronyms.....	12
8.2. Dependencies.....	12
8.3. Errata.....	12
8.4. Module History.....	12
9. Examples for GPIO.....	14
10. Quick Start Guide for the GPIO driver.....	15
10.1. Use Cases.....	15
10.2. GPIO Interrupt Usage.....	15
10.3. Setup Steps.....	15
10.3.1. Prerequisites.....	15
11. Example.....	16
11.1. Interrupt Usage.....	16
12. Document Revision History.....	17

1. Software License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.
4. This software may only be redistributed and used in connection with an Atmel microcontroller product.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

2. Prerequisites

The GPIO module depends on the following modules:

- System Clock Management (Sysclock) for clock control
- Global interrupt management for enabling or disabling interrupts
- Common IOPORT API for basic GPIO functions

3. Module Overview

The General Purpose Input/Output Controller (GPIO) controls the I/O pins of the microcontroller. Each GPIO pin may be used as a general-purpose I/O or be assigned to a function of an embedded peripheral.

The GPIO is configured using the Peripheral Bus (PB).

4. Special Considerations

There are no special considerations for this module.

5. Extra Information

For extra information, see [Extra Information for GPIO](#). This includes:

- [Acronyms](#)
- [Dependencies](#)
- [Errata](#)
- [Module History](#)

6. Examples

For a list of examples related to this driver, see [Examples for GPIO](#).

7. API Overview

7.1. Variable and Type Definitions

7.1.1. Type `gpio_pin_callback_t`

```
typedef void(* gpio_pin_callback_t )(void)
```

The interrupt handler can be configured to do a function callback, the callback function must match the `gpio_pin_callback_t` type.

7.2. Function Definitions

7.2.1. Interrupt Support

The GPIO can be configured to generate an interrupt when it detects a change on a GPIO pin.

7.2.1.1. Function `gpio_set_pin_callback()`

Set callback for given GPIO pin.

```
bool gpio_set_pin_callback(  
    ioport_pin_t pin,  
    gpio_pin_callback_t callback,  
    uint8_t irq_level)
```

Table 7-1 Parameters

Data direction	Parameter name	Description
[in]	pin	The pin number
[in]	callback	callback function pointer
[in]	irq_level	interrupt level

Table 7-2 Return Values

Return value	Description
true	Set successfully
false	Wrong parameters or maximum number of interrupt sources has been exceeding

7.2.1.2. Function `gpio_enable_pin_interrupt()`

Enable the interrupt of a pin.

```
void gpio_enable_pin_interrupt(  
    ioport_pin_t pin)
```

Table 7-3 Parameters

Data direction	Parameter name	Description
[in]	pin	The pin number

7.2.1.3. Function `gpio_disable_pin_interrupt()`

Disable the interrupt of a pin.

```
void gpio_disable_pin_interrupt(
    ioport_pin_t pin)
```

Table 7-4 Parameters

Data direction	Parameter name	Description
[in]	pin	The pin number

7.2.1.4. Function `gpio_get_pin_interrupt_flag()`

Get the interrupt flag of a pin.

```
uint32_t gpio_get_pin_interrupt_flag(
    ioport_pin_t pin)
```

Table 7-5 Parameters

Data direction	Parameter name	Description
[in]	pin	The pin number

Table 7-6 Return Values

Return value	Description
uint32_t	The pin interrupt flag (0/1)

7.2.1.5. Function `gpio_clear_pin_interrupt_flag()`

Clear the interrupt flag of a pin.

```
void gpio_clear_pin_interrupt_flag(
    ioport_pin_t pin)
```

Table 7-7 Parameters

Data direction	Parameter name	Description
[in]	pin	The pin number

7.2.2. Peripheral Event System Support

The GPIO can be programmed to output peripheral events whenever an interrupt condition is detected, such as pin value change, or only when a rising or falling edge is detected.

7.2.2.1. Function `gpio_enable_pin_periph_event()`

Enable the peripheral event generation of a pin.

```
void gpio_enable_pin_periph_event(  
    ioport_pin_t pin)
```

Table 7-8 Parameters

Data direction	Parameter name	Description
[in]	pin	The pin number

7.2.2.2. Function `gpio_disable_pin_periph_event()`

Disable the peripheral event generation of a pin.

```
void gpio_disable_pin_periph_event(  
    ioport_pin_t pin)
```

Table 7-9 Parameters

Data direction	Parameter name	Description
[in]	pin	The pin number

8. Extra Information for GPIO

8.1. Acronyms

Below is a table listing the acronyms used in this module, along with their intended meanings.

Acronym	Definition
PB	Peripheral Bus
QSG	Quick Start Guide

8.2. Dependencies

This driver has the following dependencies:

- Power Management

If the CPU enters a sleep mode that disables clocks used by the GPIO, the GPIO will stop functioning and resume operation after the system wakes up from sleep mode. If a peripheral function is configured for a GPIO pin, the peripheral will be able to control the GPIO pin even if the GPIO clock is stopped.

- Clocks

The GPIO is connected to a Peripheral Bus clock (CLK_GPIO). This clock is generated by the Power Manager. CLK_GPIO is enabled at reset, and can be disabled by writing to the Power Manager. CLK_GPIO must be enabled in order to access the configuration registers of the GPIO or to use the GPIO interrupts. After configuring the GPIO, the CLK_GPIO can be disabled by writing to the Power Manager if interrupts are not used.

- Interrupts

The GPIO interrupt request lines are connected to the NVIC. Using the GPIO interrupts requires the NVIC to be programmed first.

- Peripheral Events

The GPIO peripheral events are connected via the Peripheral Event System.

- Debug Operation

When an external debugger forces the CPU into debug mode, the GPIO continues normal operation. If the GPIO is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

8.3. Errata

There are no errata related to this driver.

8.4. Module History

An overview of the module history is presented in the table below, with details on the enhancements and fixes made to the module since its first release. The current version of this corresponds to the newest version in the table.

Changelog

Initial document release

9. Examples for GPIO

This is a list of the available Quick Start guides (QSGs) and example applications for [SAM4L General-Purpose Input/Output \(GPIO\) Driver](#). QSGs are simple examples with step-by-step instructions to configure and use this driver in a selection of use cases. Note that a QSG can be compiled as a standalone application or be added to the user application.

- [Example](#)

10. Quick Start Guide for the GPIO driver

This is the quick start guide for the `group_sam_drivers_gpio`, with step-by-step instructions on how to configure and use the driver for a specific use case.

The use cases contain several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

10.1. Use Cases

- [GPIO Interrupt Usage](#)

10.2. GPIO Interrupt Usage

This use case will demonstrate how to configure a pin(PC03) to trigger an interrupt on SAM4L-EK board.

10.3. Setup Steps

10.3.1. Prerequisites

This module requires the following service:

- Clock management
- Common IOPORT (for GPIO)

11. Example

Add this to the main loop or a setup function:

```
/* Configure push button 0 to trigger an interrupt on falling edge */
ioport_set_pin_dir(EXAMPLE_BUTTON_INT, IOPORT_DIR_INPUT);
ioport_set_pin_mode(EXAMPLE_BUTTON_INT, IOPORT_MODE_PULLUP |
    IOPORT_MODE_GLITCH_FILTER);
ioport_set_pin_sense_mode(EXAMPLE_BUTTON_INT, IOPORT_SENSE_FALLING);
if (!gpio_set_pin_callback(EXAMPLE_BUTTON_INT, pb0_callback, 1)) {
    printf("Set pin callback failure!\r\n");
    while (1) {
    }
}
gpio_enable_pin_interrupt(EXAMPLE_BUTTON_INT);
```

1. Initialize a pin to trigger an interrupt. Here, we initialize PC03 as an input pin with pull up and glitch filter and to generate an interrupt on a falling edge.

```
ioport_set_pin_dir(EXAMPLE_BUTTON_INT, IOPORT_DIR_INPUT);
ioport_set_pin_mode(EXAMPLE_BUTTON_INT, IOPORT_MODE_PULLUP |
    IOPORT_MODE_GLITCH_FILTER);
ioport_set_pin_sense_mode(EXAMPLE_BUTTON_INT, IOPORT_SENSE_FALLING);
```

2. Set a callback for the pin interrupt.

```
if (!gpio_set_pin_callback(EXAMPLE_BUTTON_INT, pb0_callback, 1)) {
    printf("Set pin callback failure!\r\n");
    while (1) {
    }
}
```

3. Enable pin interrupt.

```
gpio_enable_pin_interrupt(EXAMPLE_BUTTON_INT);
```

11.1. Interrupt Usage

When an interrupt happens on a pin, it will execute your callback function.

```
static void pb0_callback(void)
{
    /* Handle pin interrupt here e.g. toggle an LED */
    LED_Toggle(LED0);
}
```

12. Document Revision History

Doc. Rev.	Date	Comments
42280B	07/2015	Updated title of application note and added list of supported devices
42280A	05/2014	Initial document release



Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA T: (+1)(408) 441.0311 F: (+1)(408) 436.4200 | www.atmel.com

© 2015 Atmel Corporation. / Rev.: Atmel-42280B-SAM4L-General-Purpose-Input-Output-GPIO-Driver_AT07903_Application Note-07/2015

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. ARM®, ARM Connected®, and others are registered trademarks of ARM Ltd. Other terms and product names may be trademarks of others.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications") without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.