SMART ARM-based Microcontrollers

AT03248: SAM D/R/L/C Port (PORT) Driver

APPLICATION NOTE

Introduction

This driver for Atmel® | SMART ARM®-based microcontrollers provides an interface for the configuration and management of the device's General Purpose Input/Output (GPIO) pin functionality, for manual pin state reading and writing.

The following peripheral is used by this module:

- PORT (GPIO Management)

The following devices can use this module:

- Atmel | SMART SAM D20/D21
- Atmel | SMART SAM R21
- Atmel | SMART SAM D09/D10/D11
- Atmel | SMART SAM L21/L22
- Atmel | SMART SAM DA1
- Atmel | SMART SAM C20/C21

The outline of this documentation is as follows:

Table of Contents

1. Software License

2. Prerequisites

There are no prerequisites for this module.

3. Module Overview

The device GPIO (PORT) module provides an interface between the user application logic and external hardware peripherals, when general pin state manipulation is required. This driver provides an easy-to-use interface to the physical pin input samplers and output drivers, so that pins can be read from or written to for general purpose external hardware control.

3.1. Driver Feature Macro Definition

Driver Feature Macro	Supported devices
FEATURE_PORT_INPUT_EVENT	SAM L21/L22/C20/C21

Note: The specific features are only available in the driver when the selected device supports those features.

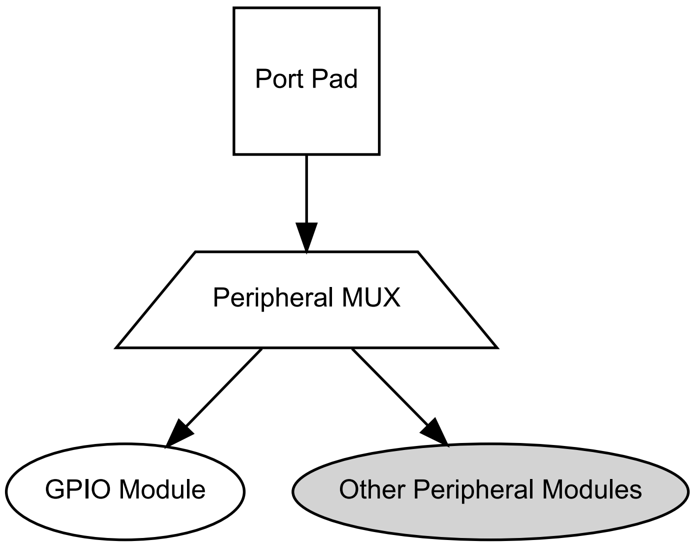
3.2. Physical and Logical GPIO Pins

SAM devices use two naming conventions for the I/O pins in the device; one physical and one logical. Each physical pin on a device package is assigned both a physical port and pin identifier (e.g. "PORTA. 0") as well as a monotonically incrementing logical GPIO number (e.g. "GPIO0"). While the former is used to map physical pins to their physical internal device module counterparts, for simplicity the design of this driver uses the logical GPIO numbers instead.

3.3. Physical Connection

Figure 3-1 Physical Connection on page 6 shows how this module is interconnected within the device.

Figure 3-1. Physical Connection



4. Special Considerations

The SAM port pin input sampler can be disabled when the pin is configured in pure output mode to save power; reading the pin state of a pin configured in output-only mode will read the logical output state that was last set.

5. Extra Information

For extra information, see Extra Information for PORT Driver. This includes:

- Acronyms
- Dependencies
- Errata
- Module History

6. Examples

For a list of examples related to this driver, see Examples for PORT Driver.

7. API Overview

7.1. Structure Definitions

7.1.1. Struct port_config

Configuration structure for a port pin instance. This structure should be initialized by the port_get_config_defaults() function before being modified by the user application.

Table 7-1. Members

Type	Name	Description
enum port_pin_dir	direction	Port buffer input/output direction
enum port_pin_pull	input_pull	Port pull-up/pull-down for input pins
bool	powersave	Enable lowest possible powerstate on the pin **Note:** All other configurations will be ignored, the pin will be disabled.

7.1.2. Struct port_input_event_config

Configuration structure for a port input event.

Table 7-2. Members

Type	Name	Description
enum port_input_event_action	action	Port input event action
uint8_t	gpio_pin	GPIO pin

7.2. Macro Definitions

7.2.1. Driver Feature Definition

Define port features set according to different device family.

7.2.1.1. Macro FEATURE_PORT_INPUT_EVENT

```
#define FEATURE_PORT_INPUT_EVENT
```

Event input control feature support for PORT group.

7.2.2. PORT Alias Macros

7.2.2.1. Macro PORTA

```
#define PORTA
```

Convenience definition for GPIO module group A on the device (if available).

7.2.2.2. Macro PORTB

```
#define PORTB
```

Convenience definition for GPIO module group B on the device (if available).

7.2.2.3. Macro PORTC

```
#define PORTC
```

Convenience definition for GPIO module group C on the device (if available).

7.2.2.4. Macro PORTD

```
#define PORTD
```

Convenience definition for GPIO module group D on the device (if available).

7.3. Function Definitions

7.3.1. State Reading/Writing (Physical Group Orientated)

7.3.1.1. Function port_get_group_from_gpio_pin()

Retrieves the PORT module group instance from a given GPIO pin number.

```
PortGroup * port_get_group_from_gpio_pin(
        const uint8_t gpio_pin)
```

Retrieves the PORT module group instance associated with a given logical GPIO pin number.

Table 7-3. Parameters

Data direction	Parameter name	Description
[in]	gpio_pin	Index of the GPIO pin to convert

Returns
Base address of the associated PORT module.

7.3.1.2. Function port_group_get_input_level()

Retrieves the state of a group of port pins that are configured as inputs.

```
uint32_t port_group_get_input_level(
        const PortGroup *const port,
        const uint32_t mask)
```

Reads the current logic level of a port module's pins and returns the current levels as a bitmask.

Table 7-4. Parameters

Data direction	Parameter name	Description
[in]	port	Base of the PORT module to read from
[in]	mask	Mask of the port pin(s) to read

Returns
Status of the port pin(s) input buffers.

7.3.1.3. Function port_group_get_output_level()

Retrieves the state of a group of port pins that are configured as outputs.

```
uint32_t port_group_get_output_level(
        const PortGroup *const port,
        const uint32_t mask)
```

Reads the current logical output level of a port module's pins and returns the current levels as a bitmask.

Table 7-5. Parameters

Data direction	Parameter name	Description
[in]	port	Base of the PORT module to read from
[in]	mask	Mask of the port pin(s) to read

Returns
Status of the port pin(s) output buffers.

7.3.1.4. Function port_group_set_output_level()

Sets the state of a group of port pins that are configured as outputs.

```
void port_group_set_output_level(
        PortGroup *const port,
        const uint32_t mask,
        const uint32_t level_mask)
```

Sets the current output level of a port module's pins to a given logic level.

Table 7-6. Parameters

Data direction	Parameter name	Description
[out]	port	Base of the PORT module to write to
[in]	mask	Mask of the port pin(s) to change
[in]	level_mask	Mask of the port level(s) to set

7.3.1.5. Function port_group_toggle_output_level()

Toggles the state of a group of port pins that are configured as an outputs.

```
void port_group_toggle_output_level(
        PortGroup *const port,
        const uint32_t mask)
```

Toggles the current output levels of a port module's pins.

Table 7-7. Parameters

Data direction	Parameter name	Description
[out]	port	Base of the PORT module to write to
[in]	mask	Mask of the port pin(s) to toggle

7.3.2. Configuration and Initialization

7.3.2.1. Function port_get_config_defaults()

Initializes a Port pin/group configuration structure to defaults.

```
void port_get_config_defaults(
        struct port_config *const config)
```

Initializes a given Port pin/group configuration structure to a set of known default values. This function should be called on all new instances of these configuration structures before being modified by the user application.

The default configuration is as follows:

- Input mode with internal pull-up enabled

Table 7-8. Parameters

Data direction	Parameter name	Description
[out]	config	Configuration structure to initialize to default values

7.3.2.2. Function port_pin_set_config()

Writes a Port pin configuration to the hardware module.

```
void port_pin_set_config(
        const uint8_t gpio_pin,
        const struct port_config *const config)
```

Writes out a given configuration of a Port pin configuration to the hardware module.

Note: If the pin direction is set as an output, the pull-up/pull-down input configuration setting is ignored.

Table 7-9. Parameters

Data direction	Parameter name	Description
[in]	gpio_pin	Index of the GPIO pin to configure
[in]	config	Configuration settings for the pin

7.3.2.3. Function port_group_set_config()

Writes a Port group configuration group to the hardware module.

```
void port_group_set_config(
        PortGroup *const port,
        const uint32_t mask,
        const struct port_config *const config)
```

Writes out a given configuration of a Port group configuration to the hardware module.

Note: If the pin direction is set as an output, the pull-up/pull-down input configuration setting is ignored.

Table 7-10. Parameters

Data direction	Parameter name	Description
[out]	port	Base of the PORT module to write to
[in]	mask	Mask of the port pin(s) to configure
[in]	config	Configuration settings for the pin group

7.3.3. State Reading/Writing (Logical Pin Orientated)

7.3.3.1. Function port_pin_get_input_level()

Retrieves the state of a port pin that is configured as an input.

```
bool port_pin_get_input_level(
        const uint8_t gpio_pin)
```

Reads the current logic level of a port pin and returns the current level as a Boolean value.

Table 7-11. Parameters

Data direction	Parameter name	Description
[in]	gpio_pin	Index of the GPIO pin to read

Returns
Status of the port pin's input buffer.

7.3.3.2. Function port_pin_get_output_level()

Retrieves the state of a port pin that is configured as an output.

```
bool port_pin_get_output_level(
        const uint8_t gpio_pin)
```

Reads the current logical output level of a port pin and returns the current level as a Boolean value.

Table 7-12. Parameters

Data direction	Parameter name	Description
[in]	gpio_pin	Index of the GPIO pin to read

Returns
Status of the port pin's output buffer.

7.3.3.3. Function port_pin_set_output_level()

Sets the state of a port pin that is configured as an output.

```
void port_pin_set_output_level(
        const uint8_t gpio_pin,
        const bool level)
```

Sets the current output level of a port pin to a given logic level.

Table 7-13. Parameters

Data direction	Parameter name	Description
[in]	gpio_pin	Index of the GPIO pin to write to
[in]	level	Logical level to set the given pin to

7.3.3.4. Function port_pin_toggle_output_level()

Toggles the state of a port pin that is configured as an output.

```
void port_pin_toggle_output_level(
        const uint8_t gpio_pin)
```

Toggles the current output level of a port pin.

Table 7-14. Parameters

Data direction	Parameter name	Description
[in]	gpio_pin	Index of the GPIO pin to toggle

7.3.4. Port Input Event

7.3.4.1. Function port_enable_input_event()

Enable the port event input.

```
enum status_code port_enable_input_event(
        const uint8_t gpio_pin,
        const enum port_input_event n)
```

Enable the port event input with the given pin and event.

Table 7-15. Parameters

Data direction	Parameter name	Description
[in]	gpio_pin	Index of the GPIO pin
[in]	n	Port input event

Table 7-16. Return Values

Return value	Description
STATUS_ERR_INVALID_ARG	Invalid parameter
STATUS_OK	Successfully

7.3.4.2. Function port_disable_input_event()

Disable the port event input.

```
enum status_code port_disable_input_event(
        const uint8_t gpio_pin,
        const enum port_input_event n)
```

Disable the port event input with the given pin and event.

Table 7-17. Parameters

Data direction	Parameter name	Description
[in]	gpio_pin	Index of the GPIO pin
[in]	gpio_pin	Port input event

Table 7-18. Return Values

Return value	Description
STATUS_ERR_INVALID_ARG	Invalid parameter
STATUS_OK	Successfully

7.3.4.3. Function port_input_event_get_config_defaults()

Retrieve the default configuration for port input event.

```
void port_input_event_get_config_defaults(
        struct port_input_event_config *const config)
```

Fills a configuration structure with the default configuration for port input event:

- Event output to pin
- Event action to be executed on PIN 0

Table 7-19. Parameters

Data direction	Parameter name	Description
[out]	config	Configuration structure to fill with default values

7.3.4.4. Function port_input_event_set_config()

Configure port input event.

```
enum status_code port_input_event_set_config(
        const enum port_input_event n,
        struct port_input_event_config *const config)
```

Configures port input event with the given configuration settings.

Table 7-20. Parameters

Data direction	Parameter name	Description
[in]	config	Port input even configuration structure containing the new config

Table 7-21. Return Values

Return value	Description
STATUS_ERR_INVALID_ARG	Invalid parameter
STATUS_OK	Successfully

7.4. Enumeration Definitions

7.4.1. Enum port_input_event

List of port input events.

Table 7-22. Members

Enum value	Description
PORT_INPUT_EVENT_0	Port input event 0
PORT_INPUT_EVENT_1	Port input event 1
PORT_INPUT_EVENT_2	Port input event 2
PORT_INPUT_EVENT_3	Port input event 3

7.4.2. Enum port_input_event_action

List of port input events action on pin.

Table 7-23. Members

Enum value	Description
PORT_INPUT_EVENT_ACTION_OUT	Event out to pin
PORT_INPUT_EVENT_ACTION_SET	Set output register of pin on event
PORT_INPUT_EVENT_ACTION_CLR	Clear output register pin on event
PORT_INPUT_EVENT_ACTION_TGL	Toggle output register pin on event

7.4.3. Enum port_pin_dir

Enum for the possible pin direction settings of the port pin configuration structure, to indicate the direction the pin should use.

Table 7-24. Members

Enum value	Description
PORT_PIN_DIR_INPUT	The pin's input buffer should be enabled, so that the pin state can be read
PORT_PIN_DIR_OUTPUT	The pin's output buffer should be enabled, so that the pin state can be set
PORT_PIN_DIR_OUTPUT_WTH_READBACK	The pin's output and input buffers should be enabled, so that the pin state can be set and read back

7.4.4. Enum port_pin_pull

Enum for the possible pin pull settings of the port pin configuration structure, to indicate the type of logic level pull the pin should use.

Table 7-25. Members

Enum value	Description
PORT_PIN_PULL_NONE	No logical pull should be applied to the pin
PORT_PIN_PULL_UP	Pin should be pulled up when idle
PORT_PIN_PULL_DOWN	Pin should be pulled down when idle

8. Extra Information for PORT Driver

8.1. Acronyms

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

Acronym	Description
GPIO	General Purpose Input/Output
MUX	Multiplexer

8.2. Dependencies

This driver has the following dependencies:

- System Pin Multiplexer Driver

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
Added input event feature
Initial release

9. Examples for PORT Driver

This is a list of the available Quick Start guides (QSGs) and example applications for SAM Port (PORT) 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.

- Quick Start Guide for PORT - Basic

9.1. Quick Start Guide for PORT - Basic

In this use case, the PORT module is configured for:

- One pin in input mode, with pull-up enabled
- One pin in output mode

This use case sets up the PORT to read the current state of a GPIO pin set as an input, and mirrors the opposite logical state on a pin configured as an output.

9.1.1. Setup

9.1.1.1. Prerequisites

There are no special setup requirements for this use-case.

9.1.1.2. Code

Copy-paste the following setup code to your user application:

```
void configure_port_pins(void)
{
    struct port_config config_port_pin;
    port_get_config_defaults(&config_port_pin);

    config_port_pin.direction  = PORT_PIN_DIR_INPUT;
    config_port_pin.input_pull = PORT_PIN_PULL_UP;
    port_pin_set_config(BUTTON_0_PIN, &config_port_pin);

    config_port_pin.direction = PORT_PIN_DIR_OUTPUT;
    port_pin_set_config(LED_0_PIN, &config_port_pin);
}
```

Add to user application initialization (typically the start of `main()`):

```
configure_port_pins();
```

9.1.1.3. Workflow

1. Create a PORT module pin configuration struct, which can be filled out to adjust the configuration of a single port pin.

   ```
   struct port_config config_port_pin;
   ```

2. Initialize the pin configuration struct with the module's default values.

   ```
   port_get_config_defaults(&config_port_pin);
   ```

   Note: This should always be performed before using the configuration struct to ensure that all values are initialized to known default settings.

3. Adjust the configuration struct to request an input pin.

```
config_port_pin.direction  = PORT_PIN_DIR_INPUT;
config_port_pin.input_pull = PORT_PIN_PULL_UP;
```

4. Configure push button pin with the initialized pin configuration struct, to enable the input sampler on the pin.

```
port_pin_set_config(BUTTON_0_PIN, &config_port_pin);
```

5. Adjust the configuration struct to request an output pin.

```
config_port_pin.direction = PORT_PIN_DIR_OUTPUT;
```

Note: The existing configuration struct may be re-used, as long as any values that have been altered from the default settings are taken into account by the user application.

6. Configure LED pin with the initialized pin configuration struct, to enable the output driver on the pin.

```
port_pin_set_config(LED_0_PIN, &config_port_pin);
```

9.1.2. Use Case

9.1.2.1. Code

Copy-paste the following code to your user application:

```
while (true) {
    bool pin_state = port_pin_get_input_level(BUTTON_0_PIN);

    port_pin_set_output_level(LED_0_PIN, !pin_state);
}
```

9.1.2.2. Workflow

1. Read in the current input sampler state of push button pin, which has been configured as an input in the use-case setup code.

```
bool pin_state = port_pin_get_input_level(BUTTON_0_PIN);
```

2. Write the inverted pin level state to LED pin, which has been configured as an output in the use-case setup code.

```
port_pin_set_output_level(LED_0_PIN, !pin_state);
```

10. Document Revision History

Doc. Rev.	Date	Comments
42113E	12/2015	Added input event feature. Added support for SAM L21/L22, SAM C21, SAM D09, and SAM DA1.
42113D	12/2014	Added support for SAM R21 and SAM D10/D11
42113C	01/2014	Added support for SAM D21
42113B	06/2013	Corrected documentation typos
42113A	06/2013	Initial document release