SMART ARM-based Microcontrollers

AT03262: SAM D/R/L/C System Pin Multiplexer (SYSTEM PINMUX) Driver

APPLICATION NOTE

Introduction

This driver for Atmel® | SMART ARM®-based microcontrollers provides an interface for the configuration and management of the device's physical I/O Pins, to alter the direction and input/drive characteristics as well as to configure the pin peripheral multiplexer selection.

The following peripheral is used by this module:

- PORT (Port I/O 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 SAM devices contain a number of General Purpose I/O pins, used to interface the user application logic and internal hardware peripherals to an external system. The Pin Multiplexer (PINMUX) driver provides a method of configuring the individual pin peripheral multiplexers to select alternate pin functions.

3.1. Driver Feature Macro Definition

Driver Feature Macro	Supported devices
FEATURE_SYSTEM_PINMUX_DRIVE_STRENGTH	SAM L21, SAM 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. Peripheral Multiplexing

SAM devices contain a peripheral MUX, which is individually controllable for each I/O pin of the device. The peripheral MUX allows you to select the function of a physical package pin - whether it will be controlled as a user controllable GPIO pin, or whether it will be connected internally to one of several peripheral modules (such as an I^2C module). When a pin is configured in GPIO mode, other peripherals connected to the same pin will be disabled.

3.4. Special Pad Characteristics

There are several special modes that can be selected on one or more I/O pins of the device, which alter the input and output characteristics of the pad.

3.4.1. Drive Strength

The Drive Strength configures the strength of the output driver on the pad. Normally, there is a fixed current limit that each I/O pin can safely drive, however some I/O pads offer a higher drive mode which increases this limit for that I/O pin at the expense of an increased power consumption.

3.4.2. Slew Rate

The Slew Rate configures the slew rate of the output driver, limiting the rate at which the pad output voltage can change with time.

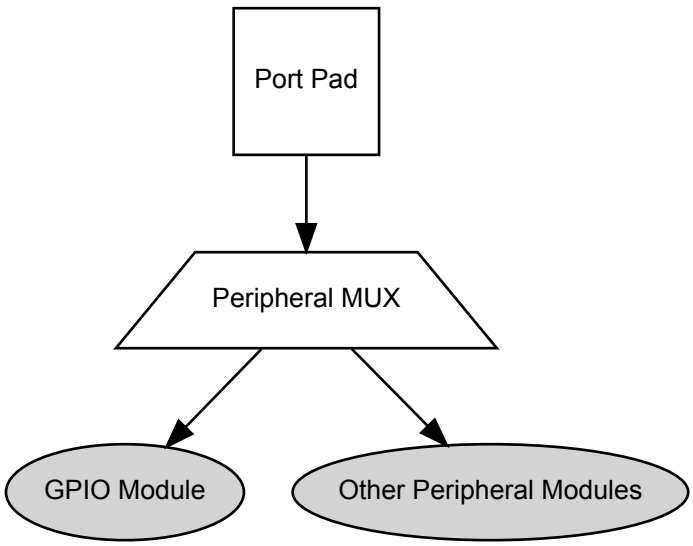
3.4.3. Input Sample Mode

The Input Sample Mode configures the input sampler buffer of the pad. By default, the input buffer is only sampled "on-demand", i.e. when the user application attempts to read from the input buffer. This mode is the most power efficient, but increases the latency of the input sample by two clock cycles of the port clock. To reduce latency, the input sampler can instead be configured to always sample the input buffer on each port clock cycle, at the expense of an increased power consumption.

3.5. Physical Connection

Figure 3-1 Physical Connection on page 7 shows how this module is interconnected within the device:

Figure 3-1. Physical Connection



4. Special Considerations

The SAM port pin input sampling mode is set in groups of four physical pins; setting the sampling mode of any pin in a sub-group of eight I/O pins will configure the sampling mode of the entire sub-group.

High Drive Strength output driver mode is not available on all device pins - refer to your device specific datasheet.

5. Extra Information

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

- Acronyms
- Dependencies
- Errata
- Module History

6. Examples

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

7. API Overview

7.1. Structure Definitions

7.1.1. Struct system_pinmux_config

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

Table 7-1. Members

Type	Name	Description
enum system_pinmux_pin_dir	direction	Port buffer input/output direction
enum system_pinmux_pin_pull	input_pull	Logic level pull of the input buffer
uint8_t	mux_position	MUX index of the peripheral that should control the pin, if peripheral control is desired. For GPIO use, this should be set to SYSTEM_PINMUX_GPIO.
bool	powersave	Enable lowest possible powerstate on the pin **Note:** All other configurations will be ignored, the pin will be disabled.

7.2. Macro Definitions

7.2.1. Macro FEATURE_SYSTEM_PINMUX_DRIVE_STRENGTH

```
#define FEATURE_SYSTEM_PINMUX_DRIVE_STRENGTH
```

Output Driver Strength Selection feature support

7.2.2. Macro SYSTEM_PINMUX_GPIO

```
#define SYSTEM_PINMUX_GPIO
```

Peripheral multiplexer index to select GPIO mode for a pin

7.3. Function Definitions

7.3.1. Configuration and Initialization

7.3.1.1. Function system_pinmux_get_config_defaults()

Initializes a Port pin configuration structure to defaults.

```
void system_pinmux_get_config_defaults(
        struct system_pinmux_config *const config)
```

Initializes a given Port pin 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:

- Non peripheral (i.e. GPIO) controlled
- Input mode with internal pull-up enabled

Table 7-2. Parameters

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

7.3.1.2. Function system_pinmux_pin_set_config()

Writes a Port pin configuration to the hardware module.

```
void system_pinmux_pin_set_config(
        const uint8_t gpio_pin,
        const struct system_pinmux_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 7-3. Parameters

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

7.3.1.3. Function system_pinmux_group_set_config()

Writes a Port pin group configuration to the hardware module.

```
void system_pinmux_group_set_config(
        PortGroup *const port,
        const uint32_t mask,
        const struct system_pinmux_config *const config)
```

Writes out a given configuration of a Port pin 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 7-4. Parameters

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

7.3.2. Special Mode Configuration (Physical Group Orientated)

7.3.2.1. Function system_pinmux_get_group_from_gpio_pin()

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

```
PortGroup * system_pinmux_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 7-5. Parameters

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

Returns
Base address of the associated PORT module.

7.3.2.2. Function system_pinmux_group_set_input_sample_mode()

Configures the input sampling mode for a group of pins.

```
void system_pinmux_group_set_input_sample_mode(
        PortGroup *const port,
        const uint32_t mask,
        const enum system_pinmux_pin_sample mode)
```

Configures the input sampling mode for a group of pins, to control when the physical I/O pin value is sampled and stored inside the microcontroller.

Table 7-6. Parameters

Data direction	Parameter name	Description
[in]	port	Base of the PORT module to configure
[in]	mask	Mask of the port pin(s) to configure
[in]	mode	New pin sampling mode to configure

7.3.3. Special Mode Configuration (Logical Pin Orientated)

7.3.3.1. Function system_pinmux_pin_get_mux_position()

Retrieves the currently selected MUX position of a logical pin.

```
uint8_t system_pinmux_pin_get_mux_position(
        const uint8_t gpio_pin)
```

Retrieves the selected MUX peripheral on a given logical GPIO pin.

Table 7-7. Parameters

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

Returns
Currently selected peripheral index on the specified pin.

7.3.3.2. Function system_pinmux_pin_set_input_sample_mode()

Configures the input sampling mode for a GPIO pin.

```
void system_pinmux_pin_set_input_sample_mode(
        const uint8_t gpio_pin,
        const enum system_pinmux_pin_sample mode)
```

Configures the input sampling mode for a GPIO input, to control when the physical I/O pin value is sampled and stored inside the microcontroller.

Table 7-8. Parameters

Data direction	Parameter name	Description
[in]	gpio_pin	Index of the GPIO pin to configure
[in]	mode	New pin sampling mode to configure

7.3.4. Function system_pinmux_group_set_output_strength()

Configures the output driver strength mode for a group of pins.

```
void system_pinmux_group_set_output_strength(
        PortGroup *const port,
        const uint32_t mask,
        const enum system_pinmux_pin_strength mode)
```

Configures the output drive strength for a group of pins, to control the amount of current the pad is able to sink/source.

Table 7-9. Parameters

Data direction	Parameter name	Description
[in]	port	Base of the PORT module to configure
[in]	mask	Mask of the port pin(s) to configure
[in]	mode	New output driver strength mode to configure

7.3.5. Function system_pinmux_pin_set_output_strength()

Configures the output driver strength mode for a GPIO pin.

```
void system_pinmux_pin_set_output_strength(
        const uint8_t gpio_pin,
        const enum system_pinmux_pin_strength mode)
```

Configures the output drive strength for a GPIO output, to control the amount of current the pad is able to sink/source.

Table 7-10. Parameters

Data direction	Parameter name	Description
[in]	gpio_pin	Index of the GPIO pin to configure
[in]	mode	New output driver strength mode to configure

7.4. Enumeration Definitions

7.4.1. Enum system_pinmux_pin_dir

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

Table 7-11. Members

Enum value	Description
SYSTEM_PINMUX_PIN_DIR_INPUT	The pin's input buffer should be enabled, so that the pin state can be read
SYSTEM_PINMUX_PIN_DIR_OUTPUT	The pin's output buffer should be enabled, so that the pin state can be set (but not read back)
SYSTEM_PINMUX_PIN_DIR_OUTPUT_WITH_READBACK	The pin's output and input buffers should both be enabled, so that the pin state can be set and read back

7.4.2. Enum system_pinmux_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 7-12. Members

Enum value	Description
SYSTEM_PINMUX_PIN_PULL_NONE	No logical pull should be applied to the pin
SYSTEM_PINMUX_PIN_PULL_UP	Pin should be pulled up when idle
SYSTEM_PINMUX_PIN_PULL_DOWN	Pin should be pulled down when idle

7.4.3. Enum system_pinmux_pin_sample

Enum for the possible input sampling modes for the port pin configuration structure, to indicate the type of sampling a port pin should use.

Table 7-13. Members

Enum value	Description
SYSTEM_PINMUX_PIN_SAMPLE_CONTINUOUS	Pin input buffer should continuously sample the pin state
SYSTEM_PINMUX_PIN_SAMPLE_ONDEMAND	Pin input buffer should be enabled when the IN register is read

7.4.4. Enum system_pinmux_pin_strength

Enum for the possible output drive strengths for the port pin configuration structure, to indicate the driver strength the pin should use.

Table 7-14. Members

Enum value	Description
SYSTEM_PINMUX_PIN_STRENGTH_NORMAL	Normal output driver strength
SYSTEM_PINMUX_PIN_STRENGTH_HIGH	High current output driver strength

8. Extra Information for SYSTEM PINMUX Driver

8.1. Acronyms

The table below presents the acronyms used in this module:

Acronym	Description
GPIO	General Purpose Input/Output
MUX	Multiplexer

8.2. Dependencies

This driver has the following dependencies:

- None

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
Removed code of open drain, slew limit and drive strength features
Fixed broken sampling mode function implementations, which wrote corrupt configuration values to the device registers
Added missing NULL pointer asserts to the PORT driver functions
Initial Release

9. Examples for SYSTEM PINMUX Driver

This is a list of the available Quick Start guides (QSGs) and example applications for SAM System Pin Multiplexer (SYSTEM PINMUX) 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 SYSTEM PINMUX - Basic

9.1. Quick Start Guide for SYSTEM PINMUX - Basic

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

- One pin in input mode, with pull-up enabled, connected to the GPIO module
- Sampling mode of the pin changed to sample on demand

This use case sets up the PINMUX to configure a physical I/O pin set as an input with pull-up and changes the sampling mode of the pin to reduce power by only sampling the physical pin state when the user application attempts to read it.

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 application:

```
struct system_pinmux_config config_pinmux;
system_pinmux_get_config_defaults(&config_pinmux);

config_pinmux.mux_position = SYSTEM_PINMUX_GPIO;
config_pinmux.direction    = SYSTEM_PINMUX_PIN_DIR_INPUT;
config_pinmux.input_pull   = SYSTEM_PINMUX_PIN_PULL_UP;

system_pinmux_pin_set_config(10, &config_pinmux);
```

9.1.1.3. Workflow

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

   ```
   struct system_pinmux_config config_pinmux;
   ```

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

   ```
   system_pinmux_get_config_defaults(&config_pinmux);
   ```

   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 with pull-up connected to the GPIO peripheral.

   ```
   config_pinmux.mux_position = SYSTEM_PINMUX_GPIO;
   config_pinmux.direction    = SYSTEM_PINMUX_PIN_DIR_INPUT;
   config_pinmux.input_pull   = SYSTEM_PINMUX_PIN_PULL_UP;
   ```

4. Configure GPIO10 with the initialized pin configuration struct, to enable the input sampler on the pin.

```
system_pinmux_pin_set_config(10, &config_pinmux);
```

9.1.2. Use Case

9.1.2.1. Code

Copy-paste the following code to your user application:

```
system_pinmux_pin_set_input_sample_mode(10,
        SYSTEM_PINMUX_PIN_SAMPLE_ONDEMAND);

while (true) {
    /* Infinite loop */
}
```

9.1.2.2. Workflow

1. Adjust the configuration of the pin to enable on-demand sampling mode.

```
system_pinmux_pin_set_input_sample_mode(10,
        SYSTEM_PINMUX_PIN_SAMPLE_ONDEMAND);
```

10. Document Revision History

Doc. Rev.	Date	Comments
42121F	12/2015	Added support for SAM L21/L22, SAM DA1, SAM D09, and SAM C20/C21
42121E	12/2014	Added support for SAM R21 and SAM D10/D11
42121D	01/2014	Added support for SAM D21
42121C	09/2013	Fixed incorrect documentation for the device pin sampling mode
42121B	06/2013	Corrected documentation typos
42121A	06/2013	Initial release

Atmel | Enabling Unlimited Possibilities®

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