SMART ARM-based Microcontrollers

AT07901: SAM4L External Interrupt Controller (EIC) Driver

APPLICATION NOTE

Introduction

This driver for Atmel® | SMART ARM®-based microcontrollers provides an interface for the configuration and management of the device's on-chip External Interrupt Controller (EIC).

Devices from the following series can use this module:

- Atmel | SMART SAM4L

The outline of this documentation is as follows:

Table of Contents

1. Software License

2. Module Overview

The External Interrupt Controller (EIC) allows pins to be configured as external interrupts. Each external interrupt has its own interrupt request and can be individually masked. Each external interrupt can generate an interrupt on rising or falling edge, or high or low level. Every interrupt input has a configurable filter to remove spikes from the interrupt source. Every interrupt pin can also be configured to be asynchronous, in order to wake-up the part from sleep modes where the CLK_SYNC clock has been disabled.

An additional Non-Maskable Interrupt (NMI) pin is also supported. This has the same properties as the other external interrupts, but is connected to the NMI request of the CPU, enabling it to interrupt any other interrupt mode.

3. Special Considerations

- The external interrupt pins (EXTINTn and NMI) may be multiplexed with I/O Controller lines. The programmer must first program the I/O Controller to assign the desired EIC pins to their peripheral function. If I/O lines of the EIC are not used by the application, they can be used for other purposes by the I/O Controller. It is only required to enable the EIC inputs actually in use. For example, if an application only requires two external interrupts, then only two I/O lines will be assigned to EIC inputs.
- All interrupts are available in all sleep modes as long as the EIC module is powered. However, in sleep modes where CLK_SYNC is stopped, the interrupt must be configured to asynchronous mode.
- The clock for the EIC bus interface (CLK_EIC) is generated by the Power Manager. This clock is enabled at reset, and can be disabled in the Power Manager. The filter and synchronous edge/level detector runs on a clock which is stopped in any of the sleep modes where the system RC oscillator (RCSYS) is not running. This clock is referred to as CLK_SYNC.
- The external interrupt request lines are connected to the NVIC. Using the external interrupts requires the NVIC to be programmed first. Using the Non-Maskable Interrupt does not require the NVIC to be programmed.
- When an external debugger forces the CPU into debug mode, the EIC continues normal operation. If the EIC 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.

4. Prerequisites

- None

5. Extra Information

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

- Acronyms
- Dependencies
- Errata
- Module History

6. Examples

For a list of examples related to this driver, see

- Quickstart guide for SAM EIC driver

7. API Overview

7.1. Variable and Type Definitions

7.1.1. Type eic_callback_t

```
typedef void(* eic_callback_t )(void)
```

7.2. Structure Definitions

7.2.1. Struct eic_line_config

Configuration parameters of the EIC module.

Table 7-1 Members

Type	Name	Description
uint8_t	eic_async	Async: EIC_ASYNCH_MODE or EIC_SYNCH_MODE
uint8_t	eic_edge	Edge: EIC_EDGE_FALLING_EDGE or EIC_EDGE_RISING_EDGE
uint8_t	eic_filter	Filter: EIC_FILTER_DISABLED or EIC_FILTER_ENABLED
uint8_t	eic_level	Level: EIC_LEVEL_LOW_LEVEL or EIC_LEVEL_HIGH_LEVEL
uint8_t	eic_mode	Mode: EIC_MODE_EDGE_TRIGGERED or EIC_MODE_LEVEL_TRIGGERED

7.3. Macro Definitions

7.3.1. External Interrupt lines

Number of available EIC lines, device dependent.

7.3.1.1. Macro EXT_NMI

```
#define EXT_NMI
```

Non-Maskable Interrupt.

7.3.1.2. Macro EXT_INT1

```
#define EXT_INT1
```

External Interrupt 1.

7.3.1.3. Macro EXT_INT2

```
#define EXT_INT2
```

External Interrupt 2.

7.3.1.4. Macro EXT_INT3

```
#define EXT_INT3
```

External Interrupt 3.

7.3.1.5. Macro EXT_INT4

```
#define EXT_INT4
```

External Interrupt 4.

7.3.1.6. Macro EXT_INT5

```
#define EXT_INT5
```

External Interrupt 5.

7.3.1.7. Macro EXT_INT6

```
#define EXT_INT6
```

External Interrupt 6.

7.3.1.8. Macro EXT_INT7

```
#define EXT_INT7
```

External Interrupt 7.

7.3.1.9. Macro EXT_INT8

```
#define EXT_INT8
```

External Interrupt 8.

7.3.2. Mode Trigger Options

7.3.2.1. Macro EIC_MODE_EDGE_TRIGGERED

```
#define EIC_MODE_EDGE_TRIGGERED
```

The interrupt is edge triggered.

7.3.2.2. Macro EIC_MODE_LEVEL_TRIGGERED

```
#define EIC_MODE_LEVEL_TRIGGERED
```

The interrupt is level triggered.

7.3.3. Edge level Options

7.3.3.1. Macro EIC_EDGE_FALLING_EDGE

```
#define EIC_EDGE_FALLING_EDGE
```

The interrupt triggers on falling edge.

7.3.3.2. Macro EIC_EDGE_RISING_EDGE

```
#define EIC_EDGE_RISING_EDGE
```

The interrupt triggers on rising edge.

7.3.4. Level Options

7.3.4.1. Macro EIC_LEVEL_LOW_LEVEL

```
#define EIC_LEVEL_LOW_LEVEL
```

The interrupt triggers on low level.

7.3.4.2. Macro EIC_LEVEL_HIGH_LEVEL

```
#define EIC_LEVEL_HIGH_LEVEL
```

The interrupt triggers on high level.

7.3.5. Filter Options

7.3.5.1. Macro EIC_FILTER_ENABLED

```
#define EIC_FILTER_ENABLED
```

The interrupt is filtered.

7.3.5.2. Macro EIC_FILTER_DISABLED

```
#define EIC_FILTER_DISABLED
```

The interrupt is not filtered.

7.3.6. Synch Mode Options

7.3.6.1. Macro EIC_SYNCH_MODE

```
#define EIC_SYNCH_MODE
```

The interrupt is synchronized to CLK_SYNC.

7.3.6.2. Macro EIC_ASYNCH_MODE

```
#define EIC_ASYNCH_MODE
```

The interrupt is asynchronous.

7.4. Function Definitions

7.4.1. Function eic_disable()

Disable the EIC module.

```
void eic_disable(
        Eic * eic)
```

Table 7-2 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC module

7.4.2. Function eic_enable()

Enable the EIC module.

```
void eic_enable(
        Eic * eic)
```

Table 7-3 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC module

7.4.3. Function eic_line_clear_interrupt()

Clear the interrupt flag of specified pin. Call this function once you have handled the interrupt.

```
void eic_line_clear_interrupt(
        Eic * eic,
        uint8_t line_number)
```

Table 7-4 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC (i.e. EIC)
[in]	line_number	Line number to clear

7.4.4. Function eic_line_disable()

Disable the external interrupt on specified line.

```
void eic_line_disable(
        Eic * eic,
        uint8_t line_number)
```

Table 7-5 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC module
[in]	line_number	Line number to disable

7.4.5. Function eic_line_disable_interrupt()

Disables the propagation from the EIC to the interrupt controller of the external interrupt on a specified line.

```
void eic_line_disable_interrupt(
        Eic * eic,
        uint8_t line_number)
```

Table 7-6 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC (i.e. EIC)
[in]	line_number	Line number of interrupt to disable

7.4.6. Function eic_line_enable()

Enable the external interrupt on specified line.

```
void eic_line_enable(
        Eic * eic,
        uint8_t line_number)
```

Table 7-7 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC module
[in]	line_number	The line number to enable

7.4.7. Function eic_line_enable_interrupt()

Enables the propagation from the EIC to the interrupt controller of the external interrupt on a specified line.

```
void eic_line_enable_interrupt(
        Eic * eic,
        uint8_t line_number)
```

Table 7-8 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC module
[in]	line_number	Line number of interrupt to enable

7.4.8. Function eic_line_interrupt_is_enabled()

Tells whether an EIC interrupt line is enabled.

```
bool eic_line_interrupt_is_enabled(
        Eic * eic,
        uint8_t line_number)
```

Table 7-9 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC module
[in]	line_number	Line number to test

Table 7-10 Return Values

Return value	Description
true	EIC interrupt line is enabled
false	EIC interrupt line is not enabled

7.4.9. Function eic_line_interrupt_is_pending()

Tells whether an EIC interrupt line is pending.

```
bool eic_line_interrupt_is_pending(
        Eic * eic,
        uint8_t line_number)
```

Table 7-11 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC module
[in]	line_number	Line number to test

Table 7-12 Return Values

Return value	Description
true	EIC interrupt line is pending
false	EIC interrupt line is not pending

7.4.10. Function eic_line_is_enabled()

Tells whether an EIC line is enabled.

```
bool eic_line_is_enabled(
        Eic * eic,
        uint8_t line_number)
```

Table 7-13 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC module
[in]	line_number	Line number to test

Table 7-14 Return Values

Return value	Description
true	EIC line is enabled
false	EIC line is not enabled

7.4.11. Function eic_line_set_callback()

Set callback for given EIC line.

```
void eic_line_set_callback(
        Eic * eic,
        uint8_t line_number,
        eic_callback_t callback,
        uint8_t irq_line,
        uint8_t irq_level)
```

Table 7-15 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC module
[in]	line_number	Number of line
[in]	callback	Callback function pointer
[in]	irq_line	Interrupt line number (EIC_1_IRQn to EIC_8_IRQn)
[in]	irq_level	Interrupt level (the priority of the interrupt request)

7.4.12. Function eic_line_set_config()

Program the EIC hardware with the specified configuration.

```
void eic_line_set_config(
        Eic * eic,
        uint8_t line_number,
        struct eic_line_config * eic_line_conf)
```

Table 7-16 Parameters

Data direction	Parameter name	Description
[in]	eic	Base address of the EIC module
[in]	line_number	Number of line to configure
[in]	eic_line_conf	Configuration parameters for the EIC module (see eic_line_config)

8. Extra Information for EIC

8.1. Acronyms

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

Acronym	Definition
EIC	External Interrupt Controller
GPIO	General-Purpose Input/Output
NMI	Non-Maskable Interrupt
NVIC	Nested Vectored Interrupt Controller
QSG	Quick Start Guide

8.2. Dependencies

- Clocks
- I/O Lines
- Interrupts
- Debug Operation

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. Quickstart guide for SAM EIC driver

This is the quickstart guide for the SAM4L External Interrupt Controller (EIC) Driver, with step-by-step instructions on how to configure and use the driver in a selection of use cases.

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.

9.1. Basic Use Case

In this basic use case, the EIC module and single line are configured for:

- Falling edge trigger and async mode
- Interrupt-based handling
- GPIO_PUSH_BUTTON_EIC_IRQ as input

9.1.1. Prerequisites

System Clock Management (Sysclock).

9.2. Setup Steps

9.2.1. Example Code

Add to application C-file:

```
static void eic_callback(void)
{
    /* Check if EIC push button line interrupt line is pending. */
    if (eic_line_interrupt_is_pending(EIC, GPIO_PUSH_BUTTON_EIC_LINE)) {
        eic_line_clear_interrupt(EIC, GPIO_PUSH_BUTTON_EIC_LINE);
        bToggle = 1;
    }
}

static void eic_setup(void)
{
    eic_enable(EIC);

    struct eic_line_config eic_line_conf;
    eic_line_conf.eic_mode = EIC_MODE_EDGE_TRIGGERED;
    eic_line_conf.eic_edge = EIC_EDGE_FALLING_EDGE;
    eic_line_conf.eic_level = EIC_LEVEL_LOW_LEVEL;
    eic_line_conf.eic_filter = EIC_FILTER_DISABLED;
    eic_line_conf.eic_async = EIC_ASYNCH_MODE;

    eic_line_set_config(EIC, GPIO_PUSH_BUTTON_EIC_LINE, &eic_line_conf);

    eic_line_set_callback(EIC, GPIO_PUSH_BUTTON_EIC_LINE, eic_callback,
    GPIO_PUSH_BUTTON_EIC_IRQ, 1);

    eic_line_enable(EIC, GPIO_PUSH_BUTTON_EIC_LINE);
}
```

9.2.2. Workflow

Define the interrupt callback function in the application:

```
static void eic_callback(void)
{
    /* Check if EIC push button line interrupt line is pending. */
    if (eic_line_interrupt_is_pending(EIC, GPIO_PUSH_BUTTON_EIC_LINE)) {
        eic_line_clear_interrupt(EIC, GPIO_PUSH_BUTTON_EIC_LINE);
        bToggle = 1;
    }
}
```

Enable EIC module:

```
eic_enable(EIC);
```

Note: Including enable module clock and lock sleep mode.

Configure EIC line with specified mode:

```
eic_line_set_config(EIC, GPIO_PUSH_BUTTON_EIC_LINE, &eic_line_conf);
```

Set the EIC callback function and enable EIC interrupt.

```
eic_line_set_callback(EIC, GPIO_PUSH_BUTTON_EIC_LINE, eic_callback,
GPIO_PUSH_BUTTON_EIC_IRQ, 1);
```

Enable EIC line:

```
eic_line_enable(EIC, GPIO_PUSH_BUTTON_EIC_LINE);
```

10. Document Revision History

Doc. Rev.	Date	Comments
42278B	07/2015	Updated title of application note and added list of supported devices
42278A	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**