SMART ARM-based Microcontrollers

AT09253: SAM4L Analog Comparator Interface Controller (ACIFC) Driver

APPLICATION NOTE

Introduction

This driver for Atmel® | SMART ARM®-based microcontrollers provides an interface for the configuration and management of the device's Analog Comparator Interface Controller functionality.

The Analog Comparator Interface controls eight Analog Comparators (AC) with identical behavior. Each Analog Comparator compares two voltages, yielding an output that depends on the result of the comparison.

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. Prerequisites

There are no prerequisites for this module.

3. Module Overview

The Analog Comparator Interface (ACIFC) controls a number of Analog Comparators (AC) with identical behavior. Each Analog Comparator compares two voltages and gives a compare output depending on the results of this comparison.

The ACIFC module can be configured in normal mode, in which each comparator is used independently, or in window mode, which uses defined comparator pairs. The comparators do not all have to be in the same mode; some comparators may be in normal mode, while others are in window mode. There are, however, restrictions on which ACs can be grouped together in window mode.

4. Special Considerations

4.1. I/O Lines

The ACIFC pins are multiplexed with other peripherals. The user application must therefore configure the GPIO Controller, to give the ACIFC module control of the associated pins.

4.2. Power Management

The ACIFC stops functioning when the system enters a sleep mode that disables its clock. However, the ACIFC module can resume its operation if the system enters the SleepWalking mode and the ACIFC clock is started by the Peripheral Event System. During this time, if the ACIFC module generates an interrupt, the system will wake-up from sleep mode and normal system operation resumes.

4.3. Clocks

The clock for ACIFC (CLK_ACIFC) is generated by the Power Manager (PM). It can be manually disabled, through the Power Manager, or may be automatically disabled when the system enters a sleep mode that disables the clocks to the peripheral bus modules.

4.4. Interrupts

The ACIFC interrupt request line is connected to the Nested Vectored Interrupt Controller (NVIC). Using the ACIFC interrupt requires that the NVIC be configured beforehand.

4.5. Peripheral Events

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

4.6. Debug Operation

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

5. Extra Information

For extra information, see Extra Information for Analog Comparator Interface Driver. This includes:

- Acronyms
- Dependencies
- Errata
- Module History

6. Examples

For a list of examples related to this driver, see Examples for Analog Comparator Interface.

7. API Overview

7.1. Variable and Type Definitions

7.1.1. Type ac_callback_t

```
typedef void(* ac_callback_t )(void)
```

ACIFC interrupt callback type.

7.1.2. Type ac_interrupt_source_t

```
typedef enum ac_interrupt_source ac_interrupt_source_t
```

ACIFC interrupt configuration type.

7.2. Structure Definitions

7.2.1. Struct ac_ch_config

ACIFC channel configuration structure.

Table 7-1 Members

Type	Name	Description
bool	always_on	Specify whether the ACIFC channel is always on
enum ac_comparator_mode	comparator_mode	The comparator mode setting
bool	event_negative	Enable/disable the output event when ACOUT is zero
bool	event_positive	Enable/disable the output event when ACOUT is one
bool	fast_mode	Enable/disable fast mode
enum ac_hysteresis_voltage	hysteresis_voltage	Hysteresis value
enum ac_ch_interrupt_setting	interrupt_setting	The interrupt setting
enum ac_negative_input	negative_input	The negative input setting

7.2.2. Struct ac_config

ACIFC module configuration structure.

Table 7-2 Members

Type	Name	Description
bool	event_trigger	Peripheral event trigger enable/disable.

7.2.3. Struct ac_dev_inst

ACIFC driver structure.

Table 7-3 Members

Type	Name	Description
struct ac_config *	cfg	Pointer to an ACIFC configuration structure.
Acifc *	hw_dev	Base address of the ACIFC module.

7.2.4. Struct ac_win_config

ACIFC window configuration structure.

Table 7-4 Members

Type	Name	Description
bool	enable	Enable/disable window mode
bool	event_enable	Enable/disable the window event from ACWOUT
enum ac_win_event_source	event_source	Window event output configuration
enum ac_win_interrupt_setting	interrupt_setting	Interrupt settings

7.3. Function Definitions

7.3.1. Function ac_ch_get_config_defaults()

Initializes an ACIFC channel configuration structure to default values.

```
void ac_ch_get_config_defaults(
        struct ac_ch_config *const cfg)
```

Note: This function should be called to prepare all new instances of ACIFC channel configuration structures before they are further modified by the user application.

The default configuration is as follows:

- Hysteresis voltage = 0mV
- ACIFC is disabled between measurements
- Fast mode is disabled
- No output peripheral event when ACOUT is zero or one
- Negative input from the external pin
- User triggered single measurement mode
- Interrupt enabled when the comparison is done

Table 7-5 Parameters

Data direction	Parameter name	Description
[out]	cfg	ACIFC channel configuration structure pointer

7.3.2. Function ac_ch_set_config()

Configure the specified ACIFC channel.

```
void ac_ch_set_config(
        struct ac_dev_inst *const dev_inst,
        uint32_t channel,
        struct ac_ch_config * cfg)
```

Table 7-6 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer
[in]	channel	ACIFC channel number (range 0 to 7 inclusive)
[in]	cfg	ACIFC channel configuration structure pointer

7.3.3. Function ac_clear_interrupt_status()

Write to the ACIFC interrupt status.

```
void ac_clear_interrupt_status(
        struct ac_dev_inst *const dev_inst,
        uint32_t status)
```

Table 7-7 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer
[in]	status	Interrupt status to be clear

7.3.4. Function ac_disable()

Disable the ACIFC Module.

```
void ac_disable(
        struct ac_dev_inst *const dev_inst)
```

Table 7-8 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

7.3.5. Function ac_disable_interrupt()

Disable the specified ACIFC interrupt source.

```
void ac_disable_interrupt(
        struct ac_dev_inst *const dev_inst,
        ac_interrupt_source_t source)
```

Table 7-9 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer
[in]	source	Interrupt source to disable

7.3.6. Function ac_enable()

Enable the ACIFC module.

```
void ac_enable(
        struct ac_dev_inst *const dev_inst)
```

Table 7-10 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

7.3.7. Function ac_enable_interrupt()

Enable the specified ACIFC interrupt source.

```
void ac_enable_interrupt(
        struct ac_dev_inst *const dev_inst,
        ac_interrupt_source_t source)
```

Table 7-11 Parameters

Data direction	Parameter name	Description
[out]	dev_inst	Driver structure pointer
[in]	source	Interrupt source to enable

7.3.8. Function ac_get_config_defaults()

Initialize an ACIFC configuration structure to default values.

```
void ac_get_config_defaults(
        struct ac_config *const cfg)
```

Note: This function should be called to prepare all new instances of ACIFC configuration structures before they are further modified by the user application.

The default configuration is as follows:

- Peripheral event trigger is disabled

Table 7-12 Parameters

Data direction	Parameter name	Description
[out]	cfg	ACIFC configuration structure pointer

7.3.9. Function ac_get_interrupt_mask()

Get the ACIFC interrupt mask.

```
uint32_t ac_get_interrupt_mask(
        struct ac_dev_inst *const dev_inst)
```

Table 7-13 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

Returns
The ACIFC interrupt mask.

7.3.10. Function ac_get_interrupt_status()

Get the ACIFC interrupt status.

```
uint32_t ac_get_interrupt_status(
        struct ac_dev_inst *const dev_inst)
```

Table 7-14 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

Returns
The ACIFC interrupt status.

7.3.11. Function ac_get_status()

Get the ACIFC status.

```
uint32_t ac_get_status(
        struct ac_dev_inst *const dev_inst)
```

Table 7-15 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

Returns
The ACIFC status.

7.3.12. Function ac_init()

Initialize the ACIFC module.

```
enum status_code ac_init(
        struct ac_dev_inst *const dev_inst,
        Acifc *const ac,
        struct ac_config *const cfg)
```

Note: Enables the ACIFC module's source clock.

Table 7-16 Parameters

Data direction	Parameter name	Description
[out]	dev_inst	Driver structure pointer
[in]	ac	Module hardware register base address pointer
[in]	cfg	Module configuration structure pointer

Returns
Status of initialization.

Table 7-17 Return Values

Return value	Description
STATUS_OK	Module initiated correctly
STATUS_ERR_DENIED	Module has already been enabled
STATUS_ERR_BUSY	Module is busy performing a comparison

7.3.13. Function ac_is_comparison_done()

Check if the ACIFC comparison is done.

```
bool ac_is_comparison_done(
        struct ac_dev_inst *const dev_inst)
```

Table 7-18 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

Returns
The ACIFC comparison result.

Table 7-19 Return Values

Return value	Description
false	ACIFC comparison has not completed
true	ACIFC comparison has completed

7.3.14. Function ac_set_callback()

Set the ACIFC interrupt callback.

```
void ac_set_callback(
        struct ac_dev_inst *const dev_inst,
        ac_interrupt_source_t source,
        ac_callback_t callback,
        uint8_t irq_level)
```

Table 7-20 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer
[in]	source	ACIFC interrupt source
[in]	callback	Interrupt callback function pointer
[in]	irq_level	Interrupt priority level

7.3.15. Function ac_user_trigger_single_comparison()

Start a single ACIFC comparison.

```
void ac_user_trigger_single_comparison(
        struct ac_dev_inst *const dev_inst)
```

Table 7-21 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

7.3.16. Function ac_win_get_config_defaults()

Initializes an ACIFC window configuration structure to default values.

```
void ac_win_get_config_defaults(
        struct ac_win_config *const cfg)
```

Note: This function should be called to prepare all new instances of ACIFC window configuration structures before they are further modified by the user application.

The default configuration is as follows:

- Window mode is disabled
- Peripheral event from ACWOUT is disabled
- Generate the window peripheral event when the comparison is done
- Window interrupt enabled when the evaluation of the common input voltage is done

Table 7-22 Parameters

Data direction	Parameter name	Description
[out]	cfg	ACIFC window configuration structure pointer

7.3.17. Function ac_win_set_config()

Configure the specified ACIFC window.

```
void ac_win_set_config(
        struct ac_dev_inst *const dev_inst,
        uint32_t window,
        struct ac_win_config * cfg)
```

Table 7-23 Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer
[in]	window	ACIFC window number (range 0 to 3 inclusive)
[in]	cfg	ACIFC window configuration structure

7.4. Enumeration Definitions

7.4.1. Enum ac_ch_interrupt_setting

ACIFC channel interrupt setting.

Table 7-24 Members

Enum value	Description
AC_CH_IS_VINP_GT_VINN	Interrupt when $V_{INP} > V_{INN}$
AC_CH_IS_VINP_LT_VINN	Interrupt when $V_{INP} < V_{INN}$
AC_CH_IS_OUTPUT_TGL	Interrupt on toggle of ACOUT
AC_CH_IS_COMP_DONE	Interrupt when comparison of V_{INP} and V_{INN} is done

7.4.2. Enum ac_comparator_mode

ACIFC comparator mode.

Table 7-25 Members

Enum value	Description
AC_COMPARATOR_OFF	Off
AC_COMPARATOR_CONTINUOUS	Continuous measurement mode
AC_COMPARATOR_USER_TRIGGERED	User triggered single measurement mode
AC_COMPARATOR_EVENT_TRIGGERED	Peripheral event triggered single measurement mode

7.4.3. Enum ac_hysteresis_voltage

ACIFC hysteresis voltage.

Table 7-26 Members

Enum value	Description
AC_HYSTERESIS_0_MV	0mV of hysteresis
AC_HYSTERESIS_25_MV	25mV of hysteresis
AC_HYSTERESIS_50_MV	50mV of hysteresis
AC_HYSTERESIS_75_MV	75mV of hysteresis

7.4.4. Enum ac_interrupt_source

ACIFC interrupt configuration type.

Table 7-27 Members

Enum value	Description
AC_INTERRUPT_CONVERSION_COMPLETED_0	Conversion completed on channel 0
AC_INTERRUPT_STARTUP_TIME_ELAPSED_0	Start-up time elapsed on channel 0
AC_INTERRUPT_CONVERSION_COMPLETED_1	Conversion completed on channel 1
AC_INTERRUPT_STARTUP_TIME_ELAPSED_1	Start-up time elapsed on channel 1
AC_INTERRUPT_CONVERSION_COMPLETED_2	Conversion completed on channel 2
AC_INTERRUPT_STARTUP_TIME_ELAPSED_2	Start-up time elapsed on channel 2
AC_INTERRUPT_CONVERSION_COMPLETED_3	Conversion completed on channel 3
AC_INTERRUPT_STARTUP_TIME_ELAPSED_3	Start-up time elapsed on channel 3
AC_INTERRUPT_CONVERSION_COMPLETED_4	Conversion completed on channel 4
AC_INTERRUPT_STARTUP_TIME_ELAPSED_4	Start-up time elapsed on channel 4
AC_INTERRUPT_CONVERSION_COMPLETED_5	Conversion completed on channel 5
AC_INTERRUPT_STARTUP_TIME_ELAPSED_5	Start-up time elapsed on channel 5
AC_INTERRUPT_CONVERSION_COMPLETED_6	Conversion completed on channel 6
AC_INTERRUPT_STARTUP_TIME_ELAPSED_6	Start-up time elapsed on channel 6
AC_INTERRUPT_CONVERSION_COMPLETED_7	Conversion completed on channel 7
AC_INTERRUPT_STARTUP_TIME_ELAPSED_7	Start-up time elapsed on channel 7
AC_INTERRUPT_WINDOW_0	Window 0 interrupt
AC_INTERRUPT_WINDOW_1	Window 1 interrupt

Enum value	Description
AC_INTERRUPT_WINDOW_2	Window 2 interrupt
AC_INTERRUPT_WINDOW_3	Window 3 interrupt

7.4.5. Enum ac_negative_input

ACIFC negative input.

Table 7-28 Members

Enum value	Description
AC_NEGTIVE_INPUT_EXTERNAL	ACANx (ACBNx) pin selected.

7.4.6. Enum ac_win_event_source

ACIFC window event output configuration.

Table 7-29 Members

Enum value	Description
AC_WIN_EVENT_ACWOUT_RISING_EDGE	ACWOUT rising edge
AC_WIN_EVENT_ACWOUT_FALLING_EDGE	ACWOUT falling edge
AC_WIN_EVENT_ACWOUT_ON_ANY_EDGE	ACWOUT rising or falling edge
AC_WIN_EVENT_INSIDE_WINDOW	Inside window
AC_WIN_EVENT_OUTSIDE_WINDOW	Outside window
AC_WIN_EVENT_MEASURE_DONE	Measurement done

7.4.7. Enum ac_win_interrupt_setting

ACIFC window interrupt settings.

Table 7-30 Members

Enum value	Description
AC_WIN_IS_VINP_INSIDE_WINDOW	Window interrupt as soon as the common input voltage is inside the window
AC_WIN_IS_VINP_OUTSIDE_WINDOW	Window interrupt as soon as the common input voltage is outside the window
AC_WIN_IS_WINDOW_OUTPUT_TGL	Window interrupt on toggle of ACWOUT
AC_WIN_IS_WINDOW_COMP_DONE	Window interrupt when evaluation of common input voltage is done

Enum value	Description
AC_WIN_IS_VINP_ENTER_WINDOW	Window interrupt when the common input voltage enters the window (e.g. the rising-edge of ACWOUT)
AC_WIN_IS_VINP_LEAVE_WINDOW	Window interrupt when the common input voltage leaves the window (e.g. the falling-edge of ACWOUT)

8. Extra Information for Analog Comparator Interface Driver

8.1. Acronyms

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

Acronym	Definition
NVIC	Nested Vectored Interrupt Controller
PES	Peripheral Event System
QSG	Quick Start Guide

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
Initial document release

9. Examples for Analog Comparator Interface

This is a list of the available Quick Start guides (QSGs) and example applications for SAM4L Analog Comparator Interface Controller (ACIFC) 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 SAM4L Analog Comparator Driver
- Analog Comparator Interface Controller - Example Using Comparison Interrupt

9.1. Quick Start Guide for SAM4L Analog Comparator Driver

This is the quick start guide for the SAM4L Analog Comparator Interface Controller (ACIFC) 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.

9.1.1. Basic Use Case

In this basic use case ACIFC channel 0 is configured to compare the inputs from ACAP0 and ACAN0.

9.1.1.1. Prerequisites

- System Clock Management (SysClock)

9.1.2. Setup Steps

9.1.2.1. Example Code

Enable the following macros in the header file conf_clock.h:

```
#define CONFIG_SYSCLK_SOURCE       SYSCLK_SRC_DFLL
```

```
#define CONFIG_DFLL0_SOURCE        GENCLK_SRC_OSC32K
```

Add the following code to your application C-file:

```
sysclk_init();
```

9.1.2.2. Workflow

1. Set system clock source as DFLL:
   ```
   #define CONFIG_SYSCLK_SOURCE       SYSCLK_SRC_DFLL
   ```
2. Set DFLL source as OSC32K:
   ```
   #define CONFIG_DFLL0_SOURCE        GENCLK_SRC_OSC32K
   ```
3. Initialize the system clock:
   ```
   sysclk_init();
   ```

9.1.3. Usage Steps

9.1.3.1. Example Code

Add the following to, for example, the main loop in your application C-file:

```
#define EXAMPLE_AC_CHANNEL         0
```

```
struct ac_dev_inst ac_device;
```

```
struct ac_config module_cfg;
ac_get_config_defaults(&module_cfg);
ac_init(&ac_device, ACIFC, &module_cfg);

ac_enable(&ac_device);

struct ac_ch_config ch_cfg;
ac_ch_get_config_defaults(&ch_cfg);
ch_cfg.always_on = true;
ch_cfg.fast_mode = true;
ac_ch_set_config(&ac_device, EXAMPLE_AC_CHANNEL, &ch_cfg);
```

```
while (!ac_is_comparison_done(&ac_device));
```

9.1.3.2. Workflow

1. Get the default configuration and initialize the module:

   ```
   struct ac_config module_cfg;
   ac_get_config_defaults(&module_cfg);
   ac_init(&ac_device, ACIFC, &module_cfg);
   ```

2. Enable the module:

   ```
   ac_enable(&ac_device);
   ```

3. Get the default configuration to initialize channel 0:

   ```
   #define EXAMPLE_AC_CHANNEL         0
   ```

   ```
   struct ac_ch_config ch_cfg;
   ac_ch_get_config_defaults(&ch_cfg);
   ch_cfg.always_on = true;
   ch_cfg.fast_mode = true;
   ac_ch_set_config(&ac_device, EXAMPLE_AC_CHANNEL, &ch_cfg);
   ```

4. Start a single comparison:

   ```
   ac_user_trigger_single_comparison(&ac_device);
   ```

5. Check if the comparison has completed:

   ```
   while (!ac_is_comparison_done(&ac_device));
   ```

9.2. Analog Comparator Interface Controller - Example Using Comparison Interrupt

9.2.1. Purpose

This example demonstrates how to use the ACIFC module and its interrupt to get a comparison result from a pair of inputs.

9.2.2. Requirements

This example has been tested on the following evaluation kits:

- SAM4L EK
- SAM4L Xplained Pro
- SAM4L8 Xplained Pro

9.2.3. Description

This example demonstrates usage of the ACIFC module. The device pins PA06 and PA07 are selected as inputs. The connection can be as follows:

For SAM4L EK

- PA06(J100.2) ADC SENSOR VBAT(J105.1)
- PA07(J4.4) VCC(J4.10)

Or

- PA06(J100.2) ADC SENSOR VBAT(J105.1)
- PA07(J4.4) GND(J4.9)

For SAM4L/SAM4L8 Xplained Pro:

- PA06(EXT3/PIN9) GND(EXT3/PIN19)
- PA07(EXT2/PIN3) VCC(EXT2/PIN20)

Or

- PA06(EXT3/PIN9) VCC(EXT3/PIN20)
- PA07(EXT2/PIN3) GND(EXT2/PIN19)

9.2.4. Main Files

- acifc.c: Analog Comparator Interface Controller driver
- acifc.h: Analog Comparator Interface Controller driver header file
- acifc_example.c: Analog Comparator Interface Controller example application

9.2.5. Compilation Information

This software is written for GNU GCC and IAR Embedded Workbench® for Atmel®. Other compilers may or may not work.

9.2.6. Usage

1. Build the program and download it into the evaluation board.
2. On the computer, open, and configure a terminal application (e.g., HyperTerminal on Microsoft® Windows®) with these settings:

- 115200 baud
- 8 bits of data
- No parity
- 1 stop bit
- No flow control

3. Start the application.
4. In the terminal window, the following text should appear:

```
* -- ACIFC IRQ Example xxx --
* -- xxxxxx-xx
* -- Compiled: xxx xx xxxx xx:xx:xx --
```

5. The application will output a different message if the voltage on pin. PA06 is lower or higher than the voltage on pin PA07 :

```
-ISR- Voltage Comparison Result: ACAP0 > ACAN0
```

or

```
-ISR- Voltage Comparison Result: ACAP0 < ACAN0
```

10. Document Revision History

Doc. Rev.	Date	Comments
42323B	05/2014	Updated title of application note and added list of supported devices
42323A	05/2014	Initial document 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**