SMART ARM-based Microcontrollers

AT10799: SAM Operational Amplifier Controller (OPAMP) Driver

APPLICATION NOTE

Introduction

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

The following peripheral is used by this module:

- OPAMP (Operational Amplifier Controller)

The following devices can use this module:

- Atmel | SMART SAM L21

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 OPAMP is an individually configurable low power, general purpose operational amplifier offering a high degree of flexibility and rail-to-rail inputs.

Each operational amplifier can be configured in standalone operational amplifier and operational amplifier with built-in feedback. All OPAMPs can be cascaded to support circuits such as differential amplifiers.
Note: For more detail configuration reference, refer to the "Built-in Modes" section in the device datasheet.

Each OPAMP has one positive and one negative input which can be flexible chosen from analog input pins including the output of another OPAMP, internal inputs such as the DAC or the resistor ladder, and the ground.

Each OPAMP output can be selected as input for AC or ADC, also available on I/O pins.

Four modes are available to select the trade-off between speed and power consumption to best fit the application requirements and optimize the power consumption.

4. Special Considerations

There are no special considerations for this module.

5. Extra Information

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

- Acronyms
- Dependencies
- Errata
- Module History

6. Examples

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

7. API Overview

7.1. Structure Definitions

7.1.1. Struct opamp0_config

Configuration structure for OPAMP 0.

Table 7-1 Members

Type	Name	Description
struct opamp_config_common	config_common	If `true`, the comparator will continue to sample during sleep mode when triggered
enum opamp0_neg_mux	negative_input	Negative input MUX selection
enum opamp0_pos_mux	positive_input	Positive input MUX selection
enum opamp0_res1_mux	r1_connection	Resistor 1 MUX selection

7.1.2. Struct opamp1_config

Configuration structure for OPAMP 1.

Table 7-2 Members

Type	Name	Description
struct opamp_config_common	config_common	If `true`, the comparator will continue to sample during sleep mode when triggered
enum opamp1_neg_mux	negative_input	Negative input MUX selection
enum opamp1_pos_mux	positive_input	Positive input MUX selection
enum opamp1_res1_mux	r1_connection	Resistor 1 MUX selection

7.1.3. Struct opamp2_config

Configuration structure for OPAMP 2.

Table 7-3 Members

Type	Name	Description
struct opamp_config_common	config_common	If `true`, the comparator will continue to sample during sleep mode when triggered
enum opamp2_neg_mux	negative_input	Negative input MUX selection
enum opamp2_pos_mux	positive_input	Positive input MUX selection
enum opamp2_res1_mux	r1_connection	Resistor 1 MUX selection

7.1.4. Struct opamp_config_common

Common configuration structure for OPAMP 0 to 2.

Table 7-4 Members

Type	Name	Description
bool	analog_out	If `true`, OPAMP output is connected to the ADC or AC input
enum opamp_bias_selection	bias_value	Bias mode selection
bool	on_demand	If `true`, the OPAMPx is enabled when a peripheral is requesting the OPAMPx to be used as an input. The OPAMPx is disabled if no peripheral is requesting it as an input.
enum opamp_pot_mux	potentiometer_selection	Potentiometer selection
bool	r1_enable	If `true`, R1 connected to RES1MUX
bool	r2_out	If `true`, resistor ladder to output
bool	r2_vcc	If `true`, resistor ladder to V_{CC}
bool	run_in_standby	If `true`, the OPAMPx is not stopped during sleep mode when triggered

7.2. Function Definitions

7.2.1. Function opamp0_get_config_defaults()

Initializes all members of OPAMP0 configuration structure to safe defaults.

```
void opamp0_get_config_defaults(
        struct opamp0_config *const config)
```

Initializes all members of OPAMP0 configuration structure to safe defaults. This function should be called on all new instances of these configuration structures before being modified by the user application.

The default configuration is setting OPAMP0 as "Voltage Follower", refer to the first mode of "Built-in Modes" in the device datasheet.

Table 7-5 Parameters

Data direction	Parameter name	Description
[out]	config	OPAMP0 configuration structure to initialize to default values

7.2.2. Function opamp0_set_config()

Writes OPAMP0 configuration to the hardware module.

```
void opamp0_set_config(
        struct opamp0_config *const config)
```

Writes a given OPAMP0 configuration to the hardware module.

Table 7-6 Parameters

Data direction	Parameter name	Description
[in]	config	Pointer to the OPAMP0 configuration struct

7.2.3. Function opamp1_get_config_defaults()

Initializes all members of OPAMP1 configuration structure to safe defaults.

```
void opamp1_get_config_defaults(
        struct opamp1_config *const config)
```

Initializes all members of OPAMP1 configuration structure to safe defaults. This function should be called on all new instances of these configuration structures before being modified by the user application.

The default configuration is setting OPAMP1 as "Voltage Follower", refer to the first mode of "Built-in Modes" in the device datasheet.

Table 7-7 Parameters

Data direction	Parameter name	Description
[out]	config	OPAMP1 configuration structure to initialize to default values

7.2.4. Function opamp1_set_config()

Writes OPAMP1 configuration to the hardware module.

```
void opamp1_set_config(
        struct opamp1_config *const config)
```

Writes a given OPAMP1 configuration to the hardware module.

Table 7-8 Parameters

Data direction	Parameter name	Description
[in]	config	Pointer to the OPAMP1 configuration struct

7.2.5. Function opamp2_get_config_defaults()

Initializes all members of OPAMP2 configuration structure to safe defaults.

```
void opamp2_get_config_defaults(
        struct opamp2_config *const config)
```

Initializes all members of OPAMP2 configuration structure to safe defaults. This function should be called on all new instances of these configuration structures before being modified by the user application.

The default configuration is setting OPAMP2 as "Voltage Follower", refer to the first mode of "Built-in Modes" in the device datasheet.

Table 7-9 Parameters

Data direction	Parameter name	Description
[out]	config	OPAMP2 configuration structure to initialize to default values

7.2.6. Function opamp2_set_config()

Writes OPAMP2 configuration to the hardware module.

```
void opamp2_set_config(
        struct opamp2_config *const config)
```

Writes a given OPAMP2 configuration to the hardware module.

Table 7-10 Parameters

Data direction	Parameter name	Description
[in]	config	Pointer to the OPAMP2 configuration struct

7.2.7. Function opamp_disable()

Disables an OPAMP that was previously enabled.

```
void opamp_disable(
        const enum opamp_id number)
```

Disables an OPAMP that was previously enabled via a call to opamp_enable().

Table 7-11 Parameters

Data direction	Parameter name	Description
[in]	number	OPAMP number to disable

7.2.8. Function opamp_enable()

Enables an OPAMP that was previously configured.

```
void opamp_enable(
        const enum opamp_id number)
```

Enables an OPAMP that was previously configured via a call to the set configuration function.

Table 7-12 Parameters

Data direction	Parameter name	Description
[in]	number	OPAMP number to enable

7.2.9. Function opamp_is_ready()

Checks an OPAMP output ready status.

```
bool opamp_is_ready(
        const enum opamp_id number)
```

Checks if an OPAMP output is ready.

Table 7-13 Parameters

Data direction	Parameter name	Description
[in]	number	OPAMP number to check

Returns
Ready status of the select OPAMP.

Table 7-14 Return Values

Return value	Description
false	If the select OPAMP output is not ready
ture	If the select OPAMP output is ready

7.2.10. Function opamp_module_disable()

Disables OPAMP module.

```
void opamp_module_disable( void )
```

Disables the peripheral.

7.2.11. Function opamp_module_enable()

Enables OPAMP module.

```
void opamp_module_enable( void )
```

Enable the peripheral. Each OPAMP must also be enabled individually by the Enable bit in the corresponding OPAMP Control register.

7.2.12. Function opamp_module_init()

Initializes OPAMP module.

```
void opamp_module_init( void )
```

Resets all registers in the MODULE to their initial state, and then enable the module.

7.2.13. Function opamp_module_reset()

Resets OPAMP module.

```
void opamp_module_reset( void )
```

Resets all registers in the MODULE to their initial state, and the OPAMP will be disabled.

7.2.14. Function opamp_voltage_doubler_disable()

Disables OPAMP voltage doubler.

```
void opamp_voltage_doubler_disable( void )
```

The analog input MUXes have high resistance, but consume less power at lower voltages (e.g., the voltage doubler is disabled).

7.2.15. Function opamp_voltage_doubler_enable()

Enables OPAMP voltage doubler.

```
void opamp_voltage_doubler_enable( void )
```

The analog input MUXes have low resistance, but consume more power at lower voltages (e.g., driven by the voltage doubler).

7.3. Enumeration Definitions

7.3.1. Enum opamp0_neg_mux

Enum for the negative input of OPAMP0.

Table 7-15 Members

Enum value	Description
OPAMP0_NEG_MUX_PIN0	Negative I/O pin 0
OPAMP0_NEG_MUX_TAP0	Resistor ladder 0 taps
OPAMP0_NEG_MUX_OUT0	OPAMP output
OPAMP0_NEG_MUX_DAC	DAC output

7.3.2. Enum opamp0_pos_mux

Enum for the positive input of OPAMP0.

Table 7-16 Members

Enum value	Description
OPAMP0_POS_MUX_PIN0	Positive I/O pin 0
OPAMP0_POS_MUX_TAP0	Resistor ladder 0 taps
OPAMP0_POS_MUX_DAC	DAC output
OPAMP0_POS_MUX_GND	Ground

7.3.3. Enum opamp0_res1_mux

Enum for the Resistor 1 of OPAMP0.

Table 7-17 Members

Enum value	Description
OPAMP0_RES1_MUX_POS_PIN0	Positive input of OPAMP0
OPAMP0_RES1_MUX_NEG_PIN0	Negative input of OPAMP0
OPAMP0_RES1_MUX_DAC	DAC output
OPAMP0_RES1_MUX_GND	Ground

7.3.4. Enum opamp1_neg_mux

Enum for the negative input of OPAMP1.

Table 7-18 Members

Enum value	Description
OPAMP1_NEG_MUX_PIN1	Negative I/O pin 1
OPAMP1_NEG_MUX_TAP1	Resistor ladder 1 taps
OPAMP1_NEG_MUX_OUT1	OPAMP output
OPAMP1_NEG_MUX_DAC	DAC output

7.3.5. Enum opamp1_pos_mux

Enum for the positive input of OPAMP1.

Table 7-19 Members

Enum value	Description
OPAMP1_POS_MUX_PIN1	Positive I/O pin 1
OPAMP1_POS_MUX_TAP1	Resistor ladder 1 taps
OPAMP1_POS_MUX_OUT0	OPAMP0 output
OPAMP1_POS_MUX_GND	Ground

7.3.6. Enum opamp1_res1_mux

Enum for the Resistor 1 of OPAMP1.

Table 7-20 Members

Enum value	Description
OPAMP1_RES1_MUX_POS_PIN0	Positive input of OPAMP1
OPAMP1_RES1_MUX_NEG_PIN0	Negative input of OPAMP1
OPAMP1_RES1_MUX_OUT0	OPAMP0 output
OPAMP1_RES1_MUX_GND	Ground

7.3.7. Enum opamp2_neg_mux

Enum for the negative input of OPAMP2.

Table 7-21 Members

Enum value	Description
OPAMP2_NEG_MUX_PIN2	Negative I/O pin 2
OPAMP2_NEG_MUX_TAP2	Resistor ladder 2 taps
OPAMP2_NEG_MUX_OUT2	OPAMP output
OPAMP2_NEG_MUX_PIN0	Negative I/O pin 0
OPAMP2_NEG_MUX_PIN1	Negative I/O pin 1
OPAMP2_NEG_MUX_DAC	DAC output

7.3.8. Enum opamp2_pos_mux

Enum for the positive input of OPAMP2.

Table 7-22 Members

Enum value	Description
OPAMP2_POS_MUX_PIN2	Positive I/O pin 2
OPAMP2_POS_MUX_TAP2	Resistor ladder 2 taps
OPAMP2_POS_MUX_OUT1	OPAMP1 output
OPAMP2_POS_MUX_GND	Ground
OPAMP2_POS_MUX_PIN0	Positive I/O pin 0
OPAMP2_POS_MUX_PIN1	Positive I/O pin 1
OPAMP2_POS_MUX_TAP0	Resistor ladder 0 taps

7.3.9. Enum opamp2_res1_mux

Enum for the Resistor 1 of OPAMP2.

Table 7-23 Members

Enum value	Description
OPAMP2_RES1_MUX_POS_PIN0	Positive input of OPAMP2
OPAMP2_RES1_MUX_NEG_PIN0	Negative input of OPAMP2
OPAMP2_RES1_MUX_OUT1	OPAMP1 output
OPAMP2_RES1_MUX_GND	Ground

7.3.10. Enum opamp_bias_selection

Enum for the Bias mode selection of OPAMP 0 to 2.

Table 7-24 Members

Enum value	Description
OPAMP_BIAS_MODE_0	Minimum current consumption but the slowest mode
OPAMP_BIAS_MODE_1	Low current consumption, slow speed
OPAMP_BIAS_MODE_2	High current consumption, fast speed
OPAMP_BIAS_MODE_3	Maximum current consumption but the fastest mode

7.3.11. Enum opamp_id

Table 7-25 Members

Enum value	Description
OPAMP_0	OPAMP 0
OPAMP_1	OPAMP 1
OPAMP_2	OPAMP 2
OPAMP_NUM	OPAMP number

7.3.12. Enum opamp_pot_mux

Enum for the potentiometer selection of OPAMP 0 to 2.

Table 7-26 Members

Enum value	Description
OPAMP_POT_MUX_14R_2R	Gain = R2/R1 = 1/7
OPAMP_POT_MUX_12R_4R	Gain = R2/R1 = 1/3
OPAMP_POT_MUX_8R_8R	Gain = R2/R1 = 1
OPAMP_POT_MUX_6R_10R	Gain = R2/R1 = 1 + 2/3
OPAMP_POT_MUX_4R_12R	Gain = R2/R1 = 3
OPAMP_POT_MUX_3R_13R	Gain = R2/R1 = 4 + 1/3
OPAMP_POT_MUX_2R_14R	Gain = R2/R1 = 7
OPAMP_POT_MUX_R_15R	Gain = R2/R1 = 15

8. Extra Information for OPAMP Driver

8.1. Acronyms

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

Acronym	Description
OPAMP	Operational Amplifier Controller

8.2. Dependencies

This driver has no dependencies.

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 Release

9. Examples for OPAMP Driver

This is a list of the available Quick Start guides (QSGs) and example applications for SAM Operational Amplifier Controller (OPAMP) 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 OPAMP - Basic

9.1. Quick Start Guide for OPAMP - Basic

In this use case, the OPAMP0 is configured as "Non-Inverting PGA" mode, refer to the second mode of "Built-in Modes" in the device datasheet.

This use case sets up the OPAMP0 to invert the input signal in OA0NEG pin, and output it to the OA0OUT pin. You can give a signal on OA0NEG and watch the output on OA0OUT through an oscilloscope.

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_non_inverting_pga_opamp0(void)
{
    struct opamp0_config conf;

    opamp_module_init();

    opamp0_get_config_defaults(&conf);

    /* Set the the OPAMP0 as "Non-Inverting PGA" mode. */
    conf.negative_input = OPAMP0_NEG_MUX_TAP0;
    conf.positive_input = OPAMP0_POS_MUX_PIN0;
    conf.r1_connection = OPAMP0_RES1_MUX_GND;
    conf.config_common.r1_enable = true;
    conf.config_common.r2_out = true;

    /* Set up OA0NEG pin and OA0OUT pin. */
    struct system_pinmux_config opamp0_neg_pin_conf;
    system_pinmux_get_config_defaults(&opamp0_neg_pin_conf);
    opamp0_neg_pin_conf.direction    = SYSTEM_PINMUX_PIN_DIR_INPUT;
    opamp0_neg_pin_conf.mux_position = MUX_PA06B_OPAMP_OAPOS0;
    system_pinmux_pin_set_config(PIN_PA06B_OPAMP_OAPOS0,
&opamp0_neg_pin_conf);
    struct system_pinmux_config opamp0_out_pin_conf;
    system_pinmux_get_config_defaults(&opamp0_out_pin_conf);
    opamp0_out_pin_conf.direction    = SYSTEM_PINMUX_PIN_DIR_OUTPUT;
    opamp0_out_pin_conf.mux_position = MUX_PA07B_OPAMP_OAOUT0;
    system_pinmux_pin_set_config(PIN_PA07B_OPAMP_OAOUT0,
&opamp0_out_pin_conf);

    opamp0_set_config(&conf);
```

```
    opamp_enable(OPAMP_0);

    /* Wait for the output ready. */
    while(!opamp_is_ready(OPAMP_0));
}
```

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

```
configure_non_inverting_pga_opamp0();
```

9.1.1.3. Workflow

1. Create an OPAMP0 configuration struct, which can be filled out to adjust the configuration of OPAMP0.

   ```
   struct opamp0_config conf;
   ```

2. Initialize the OPAMP module.

   ```
   opamp_module_init();
   ```

3. Initialize the OPAMP0 configuration struct with the module's default values.

   ```
   opamp0_get_config_defaults(&conf);
   ```

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

4. Adjust the configuration struct to set the OPAMP0 as "Non-Inverting PGA" mode.

   ```
   conf.negative_input = OPAMP0_NEG_MUX_TAP0;
   conf.positive_input = OPAMP0_POS_MUX_PIN0;
   conf.r1_connection = OPAMP0_RES1_MUX_GND;
   conf.config_common.r1_enable = true;
   conf.config_common.r2_out = true;
   ```

   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.

5. Set up OA0NEG pin and OA0OUT pin.

   ```
   struct system_pinmux_config opamp0_neg_pin_conf;
   system_pinmux_get_config_defaults(&opamp0_neg_pin_conf);
   opamp0_neg_pin_conf.direction    = SYSTEM_PINMUX_PIN_DIR_INPUT;
   opamp0_neg_pin_conf.mux_position = MUX_PA06B_OPAMP_OAPOS0;
   system_pinmux_pin_set_config(PIN_PA06B_OPAMP_OAPOS0,
   &opamp0_neg_pin_conf);
   struct system_pinmux_config opamp0_out_pin_conf;
   system_pinmux_get_config_defaults(&opamp0_out_pin_conf);
   opamp0_out_pin_conf.direction    = SYSTEM_PINMUX_PIN_DIR_OUTPUT;
   opamp0_out_pin_conf.mux_position = MUX_PA07B_OPAMP_OAOUT0;
   system_pinmux_pin_set_config(PIN_PA07B_OPAMP_OAOUT0,
   &opamp0_out_pin_conf);
   ```

6. Write OPAMP0 configuration to the hardware module.

   ```
   opamp0_set_config(&conf);
   ```

7. Enable OPAMP0.

   ```
   opamp_enable(OPAMP_0);
   ```

8. Wait for the output ready.

   ```
   while(!opamp_is_ready(OPAMP_0));
   ```

9.1.2. Use Case

9.1.2.1. Code

Copy-paste the following code to your user application:

```
while (true) {
    /* Do nothing */
}
```

10. Document Revision History

Doc. Rev.	Date	Comments
42446A	07/2015	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**