APPLICATION NOTE

AT07900: SAM4 Digital-to-Analog Converter Controller (DACC)

ASF PROGRAMMERS MANUAL

SAM4 Digital-to-Analog Converter Controller (DACC)

This driver for SAM4 (and SAM3) devices provides an interface for the configuration and management of the device's Digital-to-Analog Converter Controller functionality.

The DACC offers up to two analog outputs, making it possible for the digital-to-analog conversion to drive up to two independent analog lines. The DACC supports 10-bit or 12-bit resolution. External triggers or free running mode are configurable. The DACC integrates a Sleep Mode and also connects with a PDC channel.

The following peripherals are used by this module:

- DACC (Digital-to-Analog Converter Controller)

The outline of this documentation is as follows:

Table of Contents

Software License

1. Prerequisites

There are no prerequisites for this module.

2. Module Overview

The Digital-to-Analog Converter Controller (DACC) offers up to two analog outputs, making it possible for the digital-to-analog conversion to drive up to two independent analog lines.

The DACC module supports either 10 or 12-bit resolution (depending on SAM4 device). Data values to be converted are written into a common register for all channels. The DACC can be configured to operate in either external trigger or free running mode.

The DACC utilizes a FIFO and can also connect with a PDC/PDCA channel, both of which help reduce processor intervention.

The user application can configure DACC timings, such as Startup Time and Refresh Period.

3. Special Considerations

3.1 Voltage Reference (VREF)

The internal conversion process of the DACC can inject transient currents into the circuitry at the device's Voltage Reference (VREF) pin. Therefore the system's Voltage Reference requires adequate decoupling in order to add stability.

3.2 Output Impedance

The DACC module is not capable of driving a signal into a circuit with a low input impedance (refer to the "Analog Outputs" section in the device's datasheet). Circuits with such characteristics will require an intermediate signal buffering stage to ensure signal quality.

4. Extra Information

For extra information, see Extra Information for Digital-to-Analog Converter Controller. This includes:

- Acronyms
- Dependencies
- Errata
- Module History

5. Examples

For a list of examples related to this driver, see Examples for Digital-to-Analog Converter Controller.

6. API Overview

6.1 Variable and Type Definitions

6.1.1 Type dacc_rc_t

```
typedef enum dacc_rc dacc_rc_t
```

6.2 Macro Definitions

6.2.1 Macro DACC_MAX_DATA

```
#define DACC_MAX_DATA
```

DAC maximum data limit (depends on DACC_RESOLUTION).

6.2.2 Macro DACC_RESOLUTION

```
#define DACC_RESOLUTION
```

DAC resolution in number of data bits.

6.3 Function Definitions

6.3.1 Function dacc_disable()

Disable DACC.

```
void dacc_disable(
  Dacc * p_dacc)
```

Table 6-1. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance

Returns DACC_RC_OK on page 18 for OK.

6.3.2 Function dacc_disable_channel()

Disable DACC channel.

```
uint32_t dacc_disable_channel(
  Dacc * p_dacc,
```

```
  uint32_t ul_channel)
```

Table 6-2. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance
?	ul_channel	The output channel to disable

Returns DACC_RC_OK on page 18 for OK.

6.3.3 Function dacc_disable_interrupt()

Disable DACC interrupts.

```
void dacc_disable_interrupt(
  Dacc * p_dacc,
  uint32_t ul_interrupt_mask)
```

Table 6-3. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance
?	ul_interrupt_mask	The interrupt mask

6.3.4 Function dacc_disable_trigger()

Disable the DACC trigger (free run mode).

```
void dacc_disable_trigger(
  Dacc * p_dacc)
```

Table 6-4. Parameters

Data direction	Parameter name	Description
[in, out]	p_dacc	Module hardware register base address pointer

6.3.5 Function dacc_enable()

Enable DACC.

```
void dacc_enable(
  Dacc * p_dacc)
```

Table 6-5. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance

6.3.6 Function dacc_enable_channel()

Enable DACC channel.

```
uint32_t dacc_enable_channel(
  Dacc * p_dacc,
  uint32_t ul_channel)
```

Table 6-6. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance
?	ul_channel	The output channel to enable

Returns DACC_RC_OK on page 18 for OK.

6.3.7 Function dacc_enable_flexible_selection()

Enable the flexible channel selection mode (TAG).

```
void dacc_enable_flexible_selection(
  Dacc * p_dacc)
```

In this mode the 2 bits, DACC_CDR[13:12] which are otherwise unused, are employed to select the channel in the same way as with the USER_SEL field. Finally, if the WORD field is set, the 2 bits, DACC_CDR[13:12] are used for channel selection of the first data and the 2 bits, DACC_CDR[29:28] for channel selection of the second data.

Table 6-7. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance

6.3.8 Function dacc_enable_interrupt()

Enable DACC interrupts.

```
void dacc_enable_interrupt(
  Dacc * p_dacc,
  uint32_t ul_interrupt_mask)
```

Table 6-8. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance
?	ul_interrupt_mask	The interrupt mask

6.3.9 Function dacc_get_analog_control()

Get the analog control value.

```
uint32_t dacc_get_analog_control(
  Dacc * p_dacc)
```

Table 6-9. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance

Returns Current setting of analog control.

6.3.10 Function dacc_get_channel_status()

Get the channel status.

```
uint32_t dacc_get_channel_status(
  Dacc * p_dacc)
```

Table 6-10. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance

Returns DACC channel status.

6.3.11 Function dacc_get_interrupt_mask()

Get the interrupt mask.

```
uint32_t dacc_get_interrupt_mask(
  Dacc * p_dacc)
```

Table 6-11. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance

Returns The interrupt mask.

6.3.12 Function dacc_get_interrupt_status()

Get the interrupt status.

```
uint32_t dacc_get_interrupt_status(
  Dacc * p_dacc)
```

Table 6-12. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance

Returns The interrupt status.

6.3.13 Function dacc_get_pdc_base()

Get PDC registers base address.

```
Pdc * dacc_get_pdc_base(
  Dacc * p_dacc)
```

Table 6-13. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance

Returns DACC PDC register base address.

6.3.14 Function dacc_get_writeprotect_status()

Get the write protect status.

```
uint32_t dacc_get_writeprotect_status(
  Dacc * p_dacc)
```

Table 6-14. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance

Returns Write protect status.

6.3.15 Function dacc_reset()

Reset the DACC, emulating a hardware reset.

```
void dacc_reset(
  Dacc * p_dacc)
```

Table 6-15. Parameters

Data direction	Parameter name	Description
[out]	p_dacc	Module hardware register base address pointer

6.3.16 Function dacc_set_analog_control()

Set the analog control value.

```
uint32_t dacc_set_analog_control(
  Dacc * p_dacc,
  uint32_t ul_analog_control)
```

Table 6-16. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance
?	ul_analog_control	Analog control configuration

Returns DACC_RC_OK on page 18 for OK.

6.3.17 Function dacc_set_channel_selection()

Disable flexible (TAG) mode and select a channel for DAC outputs.

```
uint32_t dacc_set_channel_selection(
  Dacc * p_dacc,
  uint32_t ul_channel)
```

Table 6-17. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance
?	ul_channel	Channel to select

Returns DACC_RC_OK on page 18 if successful.

6.3.18 Function dacc_set_power_save()

Set the power save mode.

```
uint32_t dacc_set_power_save(
  Dacc * p_dacc,
  uint32_t ul_sleep_mode,
  uint32_t ul_fast_wakeup_mode)
```

Table 6-18. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance
?	ul_sleep_mode	Sleep mode configuration
?	ul_fast_wakeup_mode	Fast wake-up mode configuration

Returns DACC_RC_OK on page 18 if successful.

6.3.19 Function dacc_set_timing()

Set the DACC timing.

```
uint32_t dacc_set_timing(
  Dacc * p_dacc,
  uint32_t ul_startup,
  uint32_t ul_clock_divider)
```

Table 6-19. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance
?	ul_startup	Startup time selection
?	ul_clock_divider	Clock divider for internal trigger

Returns DACC_RC_OK on page 18 for OK.

6.3.20 Function dacc_set_timing()

Set DACC timings.

```
uint32_t dacc_set_timing(
  Dacc * p_dacc,
  uint32_t ul_refresh,
  uint32_t ul_maxs,
  uint32_t ul_startup)
```

Table 6-20. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance
?	ul_refresh	Refresh period setting value
?	ul_maxs	Max speed mode configuration
?	ul_startup	Startup time selection

Returns DACC_RC_OK on page 18 for OK.

6.3.21 Function dacc_set_transfer_mode()

Set the DACC word size transfer mode.

```
uint32_t dacc_set_transfer_mode(
```

```
  Dacc * p_dacc,
  uint32_t ul_mode)
```

Table 6-21. Parameters

Data direction	Parameter name	Description
[in, out]	p_dacc	Module hardware register base address pointer
[in]	ul_mode	Transfer mode configuration (1 for 32 bit, 0 for 16 bit)

Returns DACC_RC_OK if successful.

6.3.22 Function dacc_set_trigger()

Enable the DACC trigger and set the trigger source.

```
uint32_t dacc_set_trigger(
  Dacc * p_dacc,
  uint32_t ul_trigger)
```

Table 6-22. Parameters

Data direction	Parameter name	Description
[in, out]	p_dacc	Module hardware register base address pointer
[in]	ul_trigger	Trigger source number

Note For more information regarding *ul_trigger* sources refer to the section entitled "TRGSEL: Trigger Selection" in the device-specific datasheet.

Returns DACC_RC_OK if successful.

6.3.23 Function dacc_set_writeprotect()

Enable or disable write protect of DACC registers.

```
void dacc_set_writeprotect(
  Dacc * p_dacc,
  uint32_t ul_enable)
```

Table 6-23. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance
?	ul_enable	1 to enable, 0 to disable

6.3.24 Function dacc_write_conversion_data()

Write data to conversion register.

```
void dacc_write_conversion_data(
  Dacc * p_dacc,
  uint32_t ul_data)
```

Note The *ul_data* could be output data or data with channel TAG when flexible mode is used.

In flexible mode the 2 bits, DACC_CDR[13:12] which are otherwise unused, are employed to select the channel in the same way as with the USER_SEL field. Finally, if the WORD field is set, the 2 bits, DACC_CDR[13:12] are used for channel selection of the first data and the 2 bits, DACC_CDR[29:28] for channel selection of the second data.

See also dacc_enable_flexible_selection()

Table 6-24. Parameters

Data direction	Parameter name	Description
?	p_dacc	Pointer to a DACC instance
?	ul_data	The data to be transferred to analog value

6.4 Enumeration Definitions

6.4.1 Enum dacc_rc

Table 6-25. Members

Enum value	Description
DACC_RC_OK	Operation OK.
DACC_RC_INVALID_PARAM	Invalid parameter.

7. Extra Information for Digital-to-Analog Converter Controller

7.1 Acronyms

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

Acronym	Definition
CDR	Conversion Data Register
DMA	Direct Memory Access
EOC	End Of Conversion
FIFO	First In First Out
PDC	Peripheral DMA Controller
PDCA	Peripheral DMA Controller (SAM4L)
QSG	Quick Start Guide
VREF	Voltage Reference

7.2 Dependencies

This driver has the following dependencies:

- None

7.3 Errata

There are no errata related to this driver.

7.4 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

8. Examples for Digital-to-Analog Converter Controller

This is a list of the available Quick Start Guides (QSGs) and example applications for SAM4 Digital-to-Analog Converter Controller (DACC). QSGs are simple examples with step-by-step instructions to configure and use this driver in a selection of use cases. Note that QSGs can be compiled as a standalone application or be added to the user application.

- Quick Start Guide for the DACC driver
- Digital-to-Analog Converter Sine Wave Example

8.1 Quick Start Guide for the DACC driver

This is the quick start guide for the SAM4 Digital-to-Analog Converter Controller (DACC), with step-by-step instructions on how to configure and use the driver for a specific use case. The code examples can be copied into e.g. the main application loop or any other function that will need to control the AST module.

8.1.1 Use Cases

- DACC Basic Usage

8.1.2 DACC Basic Usage

This use case will demonstrate how to initialize the DACC module in half word mode.

8.1.3 Setup Steps

8.1.3.1 Prerequisites

This module requires the following service:

- System Clock Management (sysclock)

8.1.3.2 Setup Code

Add this to the main loop or a setup function:

```
```

8.1.3.3 Workflow

1. Reset the DACC module:

```
```

2. Set half word transfer mode:

```
```

3. Set the timing, sleep mode (device specific) and enable the channel:

```
```

8.1.4 Usage Steps

8.1.4.1 Usage Code

We can get the DACC status by:

```
```

```
```

We can check the DACC is ready for new data by:

```

}
```

We can update the DACC data by:

```
uint32_t dac_val = 1000;
```

8.2 Digital-to-Analog Converter Sine Wave Example

8.2.1 Purpose

This example demonstrates how to use the DACC driver to generate a sine wave.

8.2.2 Requirements

This example can be used with SAM4 evaluation kits such as the SAM4S Xplained, the SAM4N Xplained Pro, and other evaluation kits. Refer to the list of kits available for the actual device on http://www.atmel.com.

8.2.3 Description

This application demonstrates how to use the DACC in free running mode.

The example allows the frequency and amplitude of a generated sine wave to be adjusted. The sine wave's frequency can be adjusted between 200Hz and 3kHz while its peak amplitude can be adjusted between 100 to 1023 (for devices with 10-bit resolution) and 4095 (for devices with 12-bit resolution).

The example can also generate a full amplitude square wave for reference.

The sine wave can be monitored by connecting an oscilloscope to the DACC channel's output pin used by the example. Refer to *dac_sinewave_example_pin_defs* in the evaluation kit specific header file conf_dacc_sinewave_example.h for the pin to monitor.

8.2.4 Main Files

- dacc.c: Digital-to-Analog Converter Controller driver
- dacc.h: Digital-to-Analog Converter Controller driver header file
- sinewave_example.c: Digital-to-Analog Converter Controller example application

8.2.5 Compilation Information

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

8.2.6 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. In the terminal window, the following text should appear (values depend on the board and chip used):

```
*     -- DAC Sinewave Example xxx --
*     -- xxxxxx-xx
*     -- Compiled: xxx xx xxxx xx:xx:xx --
*     ======== Menu Choices for this example ========
*     -- 0: Set frequency (200Hz-3kHz).
*     -- 1: Set amplitude (100-4095).
*     -- i: Display present frequency and amplitude.
*     -- w: Switch to full amplitude square wave or back.
*     -- m: Display this menu.
*     ------------ Current configuration ------------
*     -- DACC channel: 0
*     -- Amplitude   : 2047
*     -- Frequency   : 1000
*     -- Wave        : SINE
*     ================================================
```

4. Use this menu to alter the waveform's characteristics.

Index

Document Revision History

Doc. Rev.	Date	Comments
42293A	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