
AT03789: SAM D10/D11/D20/D21/DA1/R Brown Out Detector (BOD) Driver

APPLICATION NOTE

Introduction

This driver for Atmel® | SMART ARM®-based microcontrollers provides an interface for the configuration and management of the device's Brown Out Detector (BOD) modules, to detect and respond to under-voltage events and take an appropriate action.

The following peripheral is used by this module:

- [SYSCTRL \(System Control\)](#)

The following devices can use this module:

- [Atmel | SMART SAM D20/D21](#)
- [Atmel | SMART SAM R21](#)
- [Atmel | SMART SAM D10/D11](#)
- [Atmel | SMART SAM DA1](#)

The outline of this documentation is as follows:

- [Prerequisites](#)
- [Module Overview](#)
- [Special Considerations](#)
- [Extra Information](#)
- [Examples](#)
- [API Overview](#)

Table of Contents

Introduction.....	1
1. Software License.....	3
2. Prerequisites.....	4
3. Module Overview.....	5
4. Special Considerations.....	6
5. Extra Information.....	7
6. Examples.....	8
7. API Overview.....	9
7.1. Structure Definitions.....	9
7.1.1. Struct bod_config.....	9
7.2. Function Definitions.....	9
7.2.1. Configuration and Initialization.....	9
7.3. Enumeration Definitions.....	12
7.3.1. Enum bod.....	12
7.3.2. Enum bod_action.....	12
7.3.3. Enum bod_mode.....	12
7.3.4. Enum bod_prescale.....	12
8. Extra Information for BOD Driver.....	14
8.1. Acronyms.....	14
8.2. Dependencies.....	14
8.3. Errata.....	14
8.4. Module History.....	14
9. Examples for BOD Driver.....	15
9.1. Quick Start Guide for BOD - Basic.....	15
9.1.1. Quick Start.....	15
9.1.2. Use Case.....	16
9.2. Application Use Case for BOD - Application.....	16
10. Document Revision History.....	17

1. Software License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.
4. This software may only be redistributed and used in connection with an Atmel microcontroller product.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

2. Prerequisites

There are no prerequisites for this module.

3. Module Overview

The SAM devices contain a number of Brown Out Detector (BOD) modules. Each BOD monitors the supply voltage for any dips that go below the set threshold for the module. In case of a BOD detection the BOD will either reset the system or raise a hardware interrupt so that a safe power-down sequence can be attempted.

4. Special Considerations

The time between a BOD interrupt being raised and a failure of the processor to continue executing (in the case of a core power failure) is system specific; care must be taken that all critical BOD detection events can complete within the amount of time available.

5. Extra Information

For extra information, see [Extra Information for BOD Driver](#). This includes:

- [Acronyms](#)
- [Dependencies](#)
- [Errata](#)
- [Module History](#)

6. Examples

For a list of examples related to this driver, see [Examples for BOD Driver](#).

7. API Overview

7.1. Structure Definitions

7.1.1. Struct `bod_config`

Configuration structure for a BOD module.

Table 7-1 Members

Type	Name	Description
enum <code>bod_action</code>	<code>action</code>	Action to perform when a low power detection is made
bool	<code>hysteresis</code>	If <code>true</code> , enables detection hysteresis
uint8_t	<code>level</code>	BOD level to trigger at (see electrical section of device datasheet)
enum <code>bod_mode</code>	<code>mode</code>	Sampling configuration mode for the BOD
enum <code>bod_prescale</code>	<code>prescaler</code>	Input sampler clock prescaler factor, to reduce the 1KHz clock from the ULP32K to lower the sampling rate of the BOD
bool	<code>run_in_standby</code>	If <code>true</code> , the BOD is kept enabled and sampled during device sleep

7.2. Function Definitions

7.2.1. Configuration and Initialization

7.2.1.1. Function `bod_get_config_defaults()`

Get default BOD configuration.

```
void bod_get_config_defaults(  
    struct bod_config *const conf)
```

The default BOD configuration is:

- Clock prescaler set to divide the input clock by two
- Continuous mode
- Reset on BOD detect
- Hysteresis enabled
- BOD level 0x12
- BOD kept enabled during device sleep

Table 7-2 Parameters

Data direction	Parameter name	Description
[out]	<code>conf</code>	BOD configuration struct to set to default settings

7.2.1.2. Function bod_set_config()

Configure a Brown Out Detector module.

```
enum status_code bod_set_config(  
    const enum bod bod_id,  
    struct bod_config *const conf)
```

Configures a given BOD module with the settings stored in the given configuration structure.

Table 7-3 Parameters

Data direction	Parameter name	Description
[in]	bod_id	BOD module to configure
[in]	conf	Configuration settings to use for the specified BOD

Table 7-4 Return Values

Return value	Description
STATUS_OK	Operation completed successfully
STATUS_ERR_INVALID_ARG	An invalid BOD was supplied
STATUS_ERR_INVALID_OPTION	The requested BOD level was outside the acceptable range

7.2.1.3. Function bod_enable()

Enables a configured BOD module.

```
enum status_code bod_enable(  
    const enum bod bod_id)
```

Enables the specified BOD module that has been previously configured.

Table 7-5 Parameters

Data direction	Parameter name	Description
[in]	bod_id	BOD module to enable

Returns

Error code indicating the status of the enable operation.

Table 7-6 Return Values

Return value	Description
STATUS_OK	If the BOD was successfully enabled
STATUS_ERR_INVALID_ARG	An invalid BOD was supplied

7.2.1.4. Function `bod_disable()`

Disables an enabled BOD module.

```
enum status_code bod_disable(  
    const enum bod bod_id)
```

Disables the specified BOD module that was previously enabled.

Table 7-7 Parameters

Data direction	Parameter name	Description
[in]	<code>bod_id</code>	BOD module to disable

Returns

Error code indicating the status of the disable operation.

Table 7-8 Return Values

Return value	Description
<code>STATUS_OK</code>	If the BOD was successfully disabled
<code>STATUS_ERR_INVALID_ARG</code>	An invalid BOD was supplied

7.2.1.5. Function `bod_is_detected()`

Checks if a specified BOD low voltage detection has occurred.

```
bool bod_is_detected(  
    const enum bod bod_id)
```

Determines if a specified BOD has detected a voltage lower than its configured threshold.

Table 7-9 Parameters

Data direction	Parameter name	Description
[in]	<code>bod_id</code>	BOD module to check

Returns

Detection status of the specified BOD.

Table 7-10 Return Values

Return value	Description
<code>true</code>	If the BOD has detected a low voltage condition
<code>false</code>	If the BOD has not detected a low voltage condition

7.2.1.6. Function `bod_clear_detected()`

Clears the low voltage detection state of a specified BOD.

```
void bod_clear_detected(  
    const enum bod bod_id)
```

Clears the low voltage condition of a specified BOD module, so that new low voltage conditions can be detected.

Table 7-11 Parameters

Data direction	Parameter name	Description
[in]	bod_id	BOD module to clear

7.3. Enumeration Definitions

7.3.1. Enum bod

List of possible BOD controllers within the device.

Table 7-12 Members

Enum value	Description
BOD_BOD33	BOD33 External I/O voltage

7.3.2. Enum bod_action

List of possible BOD actions when a BOD module detects a brown out condition.

Table 7-13 Members

Enum value	Description
BOD_ACTION_NONE	A BOD detect will do nothing, and the BOD state can't be polled
BOD_ACTION_RESET	A BOD detect will reset the device
BOD_ACTION_INTERRUPT	A BOD detect will fire an interrupt

7.3.3. Enum bod_mode

List of possible BOD module voltage sampling modes.

Table 7-14 Members

Enum value	Description
BOD_MODE_CONTINUOUS	BOD will sample the supply line continuously
BOD_MODE_SAMPLED	BOD will use the BOD sampling clock (1KHz) to sample the supply line

7.3.4. Enum bod_prescale

List of possible BOD controller prescaler values, to reduce the sampling speed of a BOD to lower the power consumption.

Table 7-15 Members

Enum value	Description
BOD_PRESCALE_DIV_2	Divide input prescaler clock by 2
BOD_PRESCALE_DIV_4	Divide input prescaler clock by 4
BOD_PRESCALE_DIV_8	Divide input prescaler clock by 8
BOD_PRESCALE_DIV_16	Divide input prescaler clock by 16
BOD_PRESCALE_DIV_32	Divide input prescaler clock by 32
BOD_PRESCALE_DIV_64	Divide input prescaler clock by 64
BOD_PRESCALE_DIV_128	Divide input prescaler clock by 128
BOD_PRESCALE_DIV_256	Divide input prescaler clock by 256
BOD_PRESCALE_DIV_512	Divide input prescaler clock by 512
BOD_PRESCALE_DIV_1024	Divide input prescaler clock by 1024
BOD_PRESCALE_DIV_2048	Divide input prescaler clock by 2048
BOD_PRESCALE_DIV_4096	Divide input prescaler clock by 4096
BOD_PRESCALE_DIV_8192	Divide input prescaler clock by 8192
BOD_PRESCALE_DIV_16384	Divide input prescaler clock by 16384
BOD_PRESCALE_DIV_32768	Divide input prescaler clock by 32768
BOD_PRESCALE_DIV_65536	Divide input prescaler clock by 65536

8. Extra Information for BOD Driver

8.1. Acronyms

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

Acronym	Definition
BOD	Brown Out Detector

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 BOD12 reference
Initial Release

9. Examples for BOD Driver

This is a list of the available Quick Start guides (QSGs) and example applications for [SAM Brown Out Detector \(BOD\) Driver](#). 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 BOD - Basic](#)
- [Application Use Case for BOD - Application](#)

9.1. Quick Start Guide for BOD - Basic

In this use case, the BOD33 will be configured with the following settings:

- Continuous sampling mode
- Prescaler setting of two
- Reset action on low voltage detect

9.1.1. Quick Start

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:

```
static void configure_bod33(void)
{
    struct bod_config config_bod33;
    bod_get_config_defaults(&config_bod33);

    bod_set_config(BOD_BOD33, &config_bod33);

    bod_enable(BOD_BOD33);
}
```

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

```
configure_bod33();
```

9.1.1.3. Workflow

1. Create a BOD module configuration struct, which can be filled out to adjust the configuration of a physical BOD peripheral.

```
struct bod_config config_bod33;
```

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

```
bod_get_config_defaults(&config_bod33);
```

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

3. Configure the BOD module with the desired settings.

```
bod_set_config(BOD_BOD33, &config_bod33);
```

4. Enable the BOD module so that it will monitor the power supply voltage.

```
bod_enable(BOD_BOD33);
```

9.1.2. Use Case

9.1.2.1. Code

Copy-paste the following code to your user application:

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

9.1.2.2. Workflow

1. Enter an infinite loop so that the BOD can continue to monitor the supply voltage level.

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

9.2. Application Use Case for BOD - Application

The preferred method of setting BOD33 levels and settings is through the fuses. When it is desirable to set it in software, see the below use case.

In this use case, a new BOD33 level might be set in SW if the clock settings are adjusted up after a battery has charged to a higher level. When the battery discharges, the chip will reset when the battery level is below SW BOD33 level. Now the chip will run at a lower clock rate and the BOD33 level from fuse. The chip should always measure the voltage before adjusting the frequency up.

10. Document Revision History

Doc. Rev.	Date	Comments
42149E	12/2015	Added support for SAM DA1
42149D	12/2014	Added support for SAM R21, and SAM D10/D11
42149C	01/2014	Added support for SAM D21
42149B	06/2013	Corrected documentation typos
42149A	06/2013	Initial release



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

© 2015 Atmel Corporation. / Rev.: Atmel-42149E-SAM-Brown-Out-Detector-(BOD)-Driver_AT03789_Application Note-12/2015

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. ARM®, ARM Connected®, and others are registered trademarks of ARM Ltd. Other terms and product names may be trademarks of others.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications") without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.