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 peripherals are used by this module:
- SUPC (Supply Controller)

The following devices can use this module:
- Atmel | SMART SAM C20/C21

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 bodvdd_config ................................................................................... 9

    7.2. Function Definitions .............................................................................................9
            7.2.1. Configuration and Initialization ...................................................................... 9

    7.3. Enumeration Definitions ...................................................................................11
            7.3.1. Enum bodvdd_action ................................................................................ 11
            7.3.2. Enum bodvdd_mode_in_active .............................................................11
            7.3.3. Enum bodvdd_mode_in_standby ..........................................................12
            7.3.4. Enum bodvdd_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 bodvdd_config

Configuration structure for a BODVDD module.

Table 7-1. Members

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum bodvdd_action</td>
<td>action</td>
<td>Action to perform when a low power detection is made</td>
</tr>
<tr>
<td>bool</td>
<td>hysteresis</td>
<td>If true, enables detection hysteresis</td>
</tr>
<tr>
<td>uint8_t</td>
<td>level</td>
<td>BODVDD level to trigger at when monitors VDD except in backup sleep mode</td>
</tr>
<tr>
<td>enum bodvdd_mode_in_active</td>
<td>mode_in_active</td>
<td>BODVDD configuration in active mode</td>
</tr>
<tr>
<td>enum bodvdd_mode_in_standby</td>
<td>mode_in_standby</td>
<td>BODVDD configuration in backup sleep mode</td>
</tr>
<tr>
<td>enum bodvdd_prescale</td>
<td>prescaler</td>
<td>Input sampler clock prescaler factor, to reduce the 1kHz clock from the ULP32K to lower the sampling rate of the BODVDD</td>
</tr>
<tr>
<td>bool</td>
<td>run_in_standby</td>
<td>If true, the BODVDD is kept enabled and sampled during standby</td>
</tr>
</tbody>
</table>

7.2. Function Definitions

7.2.1. Configuration and Initialization

7.2.1.1. Function bodvdd_get_config_defaults()

Get default BODVDD configuration.

```c
void bodvdd_get_config_defaults(
    struct Bodvdd_config *const conf)
```

The default BODVDD configuration is:
- Clock prescaler set to divide the input clock by two
- Continuous in active mode
- Continuous in standby mode
- Reset on BODVDD detect
- Hysteresis enabled
- BODVDD level 42 on V_DD
- BODVDD kept enabled during standby

Atmel AT12198: SAM C Brown Out Detector (BOD) Driver [APPLICATION NOTE]
Table 7-2. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[out]</td>
<td>conf</td>
<td>BODVDD configuration struct to set to default settings</td>
</tr>
</tbody>
</table>

7.2.1.2. Function bodvdd_set_config()

Configure a Brown Out Detector module.

```c
enum status_code bodvdd_set_config(
        struct bodvdd_config *const conf)
```

Table 7-3. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in]</td>
<td>conf</td>
<td>Configuration settings to use for the specified BODVDD</td>
</tr>
</tbody>
</table>

Table 7-4. Return Values

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS_OK</td>
<td>Operation completed successfully</td>
</tr>
<tr>
<td>STATUS_ERR_INVALID_ARG</td>
<td>An invalid BOD was supplied</td>
</tr>
<tr>
<td>STATUS_ERR_INVALID_OPTION</td>
<td>The requested BOD level was outside the acceptable range</td>
</tr>
</tbody>
</table>

7.2.1.3. Function bodvdd_enable()

Enables a configured BODVDD module.

```c
enum status_code bodvdd_enable( void )
```

Table 7-5. Return Values

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS_OK</td>
<td>If the BODVDD was successfully enabled</td>
</tr>
</tbody>
</table>

7.2.1.4. Function bodvdd_disable()

Disables an enabled BODVDD module.

```c
enum status_code bodvdd_disable( void )
```

Table 7-5. Return Values

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS_OK</td>
<td>If the BODVDD was successfully enabled</td>
</tr>
</tbody>
</table>
Table 7-6. Return Values

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS_OK</td>
<td>If the BODVDD was successfully disabled</td>
</tr>
</tbody>
</table>

7.2.1.5. Function bodvdd_is_detected()

Checks if the BODVDD low voltage detection has occurred.

```c
bool bodvdd_is_detected( void )
```

Determines if the BODVDD has detected a voltage lower than its configured threshold.

**Returns**

Detection status of the BODVDD.

Table 7-7. Return Values

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>If the BODVDD has detected a low voltage condition</td>
</tr>
<tr>
<td>false</td>
<td>If the BODVDD has not detected a low voltage condition</td>
</tr>
</tbody>
</table>

7.2.1.6. Function bodvdd_clear_detected()

Clears the low voltage detection state of the BODVDD.

```c
void bodvdd_clear_detected( void )
```

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

7.3. Enumeration Definitions

7.3.1. Enum bodvdd_action

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

Table 7-8. Members

<table>
<thead>
<tr>
<th>Enum value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BODVDD_ACTION_NONE</td>
<td>A BODVDD detect will do nothing, and the BODVDD state must be polled</td>
</tr>
<tr>
<td>BODVDD_ACTION_RESET</td>
<td>A BODVDD detect will reset the device</td>
</tr>
<tr>
<td>BODVDD_ACTION_INTERRUPT</td>
<td>A BODVDD detect will fire an interrupt</td>
</tr>
<tr>
<td>BODVDD_ACTION_BACKUP</td>
<td>A BODVDD detect will put the device in backup sleep mode</td>
</tr>
</tbody>
</table>

7.3.2. Enum bodvdd_mode_in_active

List of possible BODVDD module voltage sampling modes in active sleep mode.
Table 7-9. Members

<table>
<thead>
<tr>
<th>Enum value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BODVDD_ACTCFG_CONTINUOUS</td>
<td>BODVDD will sample the supply line continuously</td>
</tr>
<tr>
<td>BODVDD_ACTCFG_SAMPLED</td>
<td>BODVDD will use the BODVDD sampling clock (1kHz) to sample the supply line</td>
</tr>
</tbody>
</table>

7.3.3. Enum bodvdd_mode_in_standby

List of possible BODVDD module voltage sampling modes in standby sleep mode.

Table 7-10. Members

<table>
<thead>
<tr>
<th>Enum value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BODVDD_STDBYCFG_CONTINUOUS</td>
<td>BODVDD will sample the supply line continuously</td>
</tr>
<tr>
<td>BODVDD_STDBYCFG_SAMPLED</td>
<td>BODVDD will use the BODVDD sampling clock (1kHz) to sample the supply line</td>
</tr>
</tbody>
</table>

7.3.4. Enum bodvdd_prescale

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

Table 7-11. Members

<table>
<thead>
<tr>
<th>Enum value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BODVDD_PRESCALE_DIV_2</td>
<td>Divide input prescaler clock by 2</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_4</td>
<td>Divide input prescaler clock by 4</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_8</td>
<td>Divide input prescaler clock by 8</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_16</td>
<td>Divide input prescaler clock by 16</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_32</td>
<td>Divide input prescaler clock by 32</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_64</td>
<td>Divide input prescaler clock by 64</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_128</td>
<td>Divide input prescaler clock by 128</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_256</td>
<td>Divide input prescaler clock by 256</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_512</td>
<td>Divide input prescaler clock by 512</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_1024</td>
<td>Divide input prescaler clock by 1024</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_2048</td>
<td>Divide input prescaler clock by 2048</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_4096</td>
<td>Divide input prescaler clock by 4096</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_8192</td>
<td>Divide input prescaler clock by 8192</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_16384</td>
<td>Divide input prescaler clock by 16384</td>
</tr>
<tr>
<td>Enum value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_32768</td>
<td>Divide input prescaler clock by 32768</td>
</tr>
<tr>
<td>BODVDD_PRESCALE_DIV_65536</td>
<td>Divide input prescaler clock by 65536</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOD</td>
<td>Brown Out Detector</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Changelog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Release</td>
</tr>
</tbody>
</table>
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 a QSG 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 BODVDD will be configured with the following settings:

- Continuous sampling mode
- Prescaler setting of 2
- 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:

```c
static void configure_bodvdd(void)
{
    struct bodvdd_config config_bodvdd;
    bodvdd_get_config_defaults(&config_bodvdd);
    bodvdd_set_config(&config_bodvdd);
    bodvdd_enable();
}
```

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

```c
configure_bodvdd();
```

9.1.1.3. **Workflow**

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

   ```c
   struct bodvdd_config config_bodvdd;
   ```

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

   ```c
   bodvdd_get_config_defaults(&config_bodvdd);
   ```

   **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 BODVDD module with the desired settings.

   ```c
   bodvdd_set_config(&config_bodvdd);
   ```
4. Enable the BODVDD module so that it will monitor the power supply voltage.

```c
bodvdd_enable();
```

9.1.2. Use Case

9.1.2.1. Code

Copy-paste the following code to your user application:

```c
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.

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

9.2. Application Use Case for BOD - Application

The preferred method of setting BODVDD 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 BODVDD level might be set in SW if the clock settings are adjusted after a battery has charged to a higher level. When the battery discharges, the chip will reset when the battery level is below the SW BODVDD level. Now the chip will run at a lower clock rate and the BODVDD level from fuse. The chip should always measure the voltage before adjusting the frequency up.
## 10. Document Revision History

<table>
<thead>
<tr>
<th>Doc. Rev.</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>42701A</td>
<td>08/2016</td>
<td>Initial document release</td>
</tr>
</tbody>
</table>