NAME

avr_sleep

SYNOPSIS

Functions

void **sleep_enable** (void)
void **sleep_disable** (void)
void **sleep_cpu** (void)
void **sleep_mode** (void)
void **sleep_bod_disable** (void)

Detailed Description

#include <avr/sleep.h>

Use of the SLEEP instruction can allow an application to reduce its power comsumption considerably. AVR devices can be put into different sleep modes. Refer to the datasheet for the details relating to the device you are using.

There are several macros provided in this header file to actually put the device into sleep mode. The simplest way is to optionally set the desired sleep mode using `set_sleep_mode()` (it usually defaults to idle mode where the CPU is put on sleep but all peripheral clocks are still running), and then call **sleep_mode(). This macro automatically sets the sleep enable bit, goes to sleep, and clears the sleep enable bit.**

Example:

```
#include <avr/sleep.h>

...
 set_sleep_mode(<mode>);
 sleep_mode();
```

Note that unless your purpose is to completely lock the CPU (until a hardware reset), interrupts need to be enabled before going to sleep.

As the **sleep_mode() macro might cause race conditions in some situations, the individual steps of manipulating the sleep enable (SE) bit, and actually issuing the** SLEEP **instruction, are provided in the macros sleep_enable(), sleep_disable(), and sleep_cpu(). This also allows for test-and-sleep scenarios that take care of not missing the interrupt that will awake the device from sleep.**

Example:

```
#include <avr/interrupt.h>
#include <avr/sleep.h>

...
 set_sleep_mode(<mode>);
 cli();
 if (some_condition)
 {
  sleep_enable();
  sei();
  sleep_cpu();
  sleep_disable();
 }
 sei();
```

This sequence ensures an atomic test of `some_condition` with interrupts being disabled. If the condition is met, sleep mode will be prepared, and the SLEEP instruction will be scheduled immediately after an SEI instruction. As the intruction right after the SEI is guaranteed to be executed before an interrupt could trigger, it is sure the device will really be put to sleep.

Some devices have the ability to disable the Brown Out Detector (BOD) before going to sleep. This

will also reduce power while sleeping. If the specific AVR device has this ability then an additional macro is defined: **sleep_bod_disable(). This macro generates inlined assembly code that will correctly implement the timed sequence for disabling the BOD before sleeping. However, there is a limited number of cycles after the BOD has been disabled that the device can be put into sleep mode, otherwise the BOD will not truly be disabled. Recommended practice is to disable the BOD (sleep_bod_disable()), set the interrupts (sei()), and then put the device to sleep (sleep_cpu()), like so:**

```
#include <avr/interrupt.h>
#include <avr/sleep.h>

...
 set_sleep_mode(<mode>);
 cli();
 if (some_condition)
 {
  sleep_enable();
  sleep_bod_disable();
  sei();
  sleep_cpu();
  sleep_disable();
 }
 sei();
```

Function Documentation

void sleep_bod_disable (void)

Disable BOD before going to sleep. Not available on all devices.

void sleep_cpu (void)

Put the device into sleep mode. The SE bit must be set beforehand, and it is recommended to clear it afterwards.

void sleep_disable (void)

Clear the SE (sleep enable) bit.

void sleep_enable (void)

Put the device in sleep mode. How the device is brought out of sleep mode depends on the specific mode selected with the set_sleep_mode() function. See the data sheet for your device for more details.

Set the SE (sleep enable) bit.

void sleep_mode (void)

Put the device into sleep mode, taking care of setting the SE bit before, and clearing it afterwards.

Author

Generated automatically by Doxygen for avr-libc from the source code.