sleep.h

Go to the documentation of this file.

```
00001 /* Copyright (c) 2002, 2004 Theodore A. Roth
00002    Copyright (c) 2004, 2007, 2008 Eric B. Weddington
00003    Copyright (c) 2005, 2006, 2007 Joerg Wunsch
00004    All rights reserved.
00005
00006    Redistribution and use in source and binary forms, with or without
00007    modification, are permitted provided that the following conditions are met:
00008
00009    * Redistributions of source code must retain the above copyright
00010      notice, this list of conditions and the following disclaimer.
00011
00012    * Redistributions in binary form must reproduce the above copyright
00013      notice, this list of conditions and the following disclaimer in
00014      the documentation and/or other materials provided with the
00015      distribution.
00016
00017    * Neither the name of the copyright holders nor the names of
00018      contributors may be used to endorse or promote products derived
00019      from this software without specific prior written permission.
00020
00021    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00022    AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00023    IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00024    ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
00025    LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00026    CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00027    SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00028    INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00029    CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00030    ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00031    POSSIBILITY OF SUCH DAMAGE. */
00032
00033 /* $Id: sleep_8h_source.html,v 1.1.1.4 2012/01/03 16:04:22 joerg_wunsch Exp $ */
00034
00035 #ifndef _AVR_SLEEP_H_
00036 #define _AVR_SLEEP_H_ 1
00037
00038 #include <avr/io.h>
00039 #include <stdint.h>
00040
00041
00042 /** \file */
00043
00044 /** \defgroup avr_sleep <avr/sleep.h>: Power Management and Sleep Modes
00045
00046     \code #include <avr/sleep.h>\endcode
00047
00048     Use of the \c SLEEP instruction can allow an application to reduce its
00049     power comsumption considerably. AVR devices can be put into different
00050     sleep modes. Refer to the datasheet for the details relating to the device
00051     you are using.
00052
00053     There are several macros provided in this header file to actually
00054     put the device into sleep mode.  The simplest way is to optionally
00055     set the desired sleep mode using \c set_sleep_mode() (it usually
00056     defaults to idle mode where the CPU is put on sleep but all
00057     peripheral clocks are still running), and then call
00058     \c sleep_mode(). This macro automatically sets the sleep enable bit, goes
00059     to sleep, and clears the sleep enable bit.
00060
00061     Example:
00062     \code
00063     #include <avr/sleep.h>
00064
00065     ...
00066       set_sleep_mode(<mode>);
00067       sleep_mode();
00068     \endcode
00069
00070     Note that unless your purpose is to completely lock the CPU (until a
00071     hardware reset), interrupts need to be enabled before going to sleep.
00072
00073     As the \c sleep_mode() macro might cause race conditions in some
00074     situations, the individual steps of manipulating the sleep enable
00075     (SE) bit, and actually issuing the \c SLEEP instruction, are provided
00076     in the macros \c sleep_enable(), \c sleep_disable(), and
```

```
00077     \c sleep_cpu().  This also allows for test-and-sleep scenarios that
00078     take care of not missing the interrupt that will awake the device
00079     from sleep.
00080
00081     Example:
00082     \code
00083     #include <avr/interrupt.h>
00084     #include <avr/sleep.h>
00085
00086     ...
00087       set_sleep_mode(<mode>);
00088       cli();
00089       if (some_condition)
00090       {
00091         sleep_enable();
00092         sei();
00093         sleep_cpu();
00094         sleep_disable();
00095       }
00096       sei();
00097     \endcode
00098
00099     This sequence ensures an atomic test of \c some_condition with
00100     interrupts being disabled.  If the condition is met, sleep mode
00101     will be prepared, and the \c SLEEP instruction will be scheduled
00102     immediately after an \c SEI instruction.  As the intruction right
00103     after the \c SEI is guaranteed to be executed before an interrupt
00104     could trigger, it is sure the device will really be put to sleep.
00105
00106     Some devices have the ability to disable the Brown Out Detector (BOD) before
00107     going to sleep. This will also reduce power while sleeping. If the
00108     specific AVR device has this ability then an additional macro is defined:
00109     \c sleep_bod_disable(). This macro generates inlined assembly code
00110     that will correctly implement the timed sequence for disabling the BOD
00111     before sleeping. However, there is a limited number of cycles after the
00112     BOD has been disabled that the device can be put into sleep mode, otherwise
00113     the BOD will not truly be disabled. Recommended practice is to disable
00114     the BOD (\c sleep_bod_disable()), set the interrupts (\c sei()), and then
00115     put the device to sleep (\c sleep_cpu()), like so:
00116
00117     \code
00118     #include <avr/interrupt.h>
00119     #include <avr/sleep.h>
00120
00121     ...
00122       set_sleep_mode(<mode>);
00123       cli();
00124       if (some_condition)
00125       {
00126         sleep_enable();
00127         sleep_bod_disable();
00128         sei();
00129         sleep_cpu();
00130         sleep_disable();
00131       }
00132       sei();
00133     \endcode
00134 */
00135
00136
00137 /* Define an internal sleep control register and an internal sleep enable bit mask. */
00138 #if defined(SLEEP_CTRL)
00139
00140     /* XMEGA devices */
00141     #define _SLEEP_CONTROL_REG  SLEEP_CTRL
00142     #define _SLEEP_ENABLE_MASK  SLEEP_SEN_bm
00143
00144 #elif defined(SMCR)
00145
00146     #define _SLEEP_CONTROL_REG  SMCR
00147     #define _SLEEP_ENABLE_MASK  _BV(SE)
00148
00149 #elif defined(__AVR_AT94K__)
00150
00151     #define _SLEEP_CONTROL_REG  MCUR
00152     #define _SLEEP_ENABLE_MASK  _BV(SE)
00153
00154 #else
00155
00156     #define _SLEEP_CONTROL_REG  MCUCR
00157     #define _SLEEP_ENABLE_MASK  _BV(SE)
00158
00159 #endif
00160
00161
00162 /* Define set_sleep_mode() and sleep mode values per device. */
00163 #if defined(__AVR_ATmega161__)
00164
00165     #define SLEEP_MODE_IDLE         0
```

```
00166     #define SLEEP_MODE_PWR_DOWN     1
00167     #define SLEEP_MODE_PWR_SAVE     2
00168
00169     #define set_sleep_mode(mode) \
00170     do { \
00171         MCUCR = ((MCUCR & ~_BV(SM1)) | ((mode) == SLEEP_MODE_PWR_DOWN || (mode) == SLEEP_MODE_PWR_SAVE ? _BV(SM1) : 0)); \
00172         EMCUCR = ((EMCUCR & ~_BV(SM0)) | ((mode) == SLEEP_MODE_PWR_SAVE ? _BV(SM0) : 0)); \
00173     } while(0)
00174
00175
00176 #elif defined(__AVR_ATmega162__) \
00177 || defined(__AVR_ATmega8515__)
00178
00179     #define SLEEP_MODE_IDLE         0
00180     #define SLEEP_MODE_PWR_DOWN     1
00181     #define SLEEP_MODE_PWR_SAVE     2
00182     #define SLEEP_MODE_ADC          3
00183     #define SLEEP_MODE_STANDBY      4
00184     #define SLEEP_MODE_EXT_STANDBY  5
00185
00186     #define set_sleep_mode(mode) \
00187     do { \
00188         MCUCR = ((MCUCR & ~_BV(SM1)) | ((mode) == SLEEP_MODE_IDLE ? 0 : _BV(SM1))); \
00189         MCUCSR = ((MCUCSR & ~_BV(SM2)) | ((mode) == SLEEP_MODE_STANDBY  || (mode) == SLEEP_MODE_EXT_STANDBY ? _BV(SM2) : 0)); \
00190         EMCUCR = ((EMCUCR & ~_BV(SM0)) | ((mode) == SLEEP_MODE_PWR_SAVE || (mode) == SLEEP_MODE_EXT_STANDBY ? _BV(SM0) : 0)); \
00191     } while(0)
00192
00193 #elif defined(__AVR_AT90S2313__) \
00194 || defined(__AVR_AT90S2323__) \
00195 || defined(__AVR_AT90S2333__) \
00196 || defined(__AVR_AT90S2343__) \
00197 || defined(__AVR_AT43USB320__) \
00198 || defined(__AVR_AT43USB355__) \
00199 || defined(__AVR_AT90S4414__) \
00200 || defined(__AVR_AT90S4433__) \
00201 || defined(__AVR_AT90S8515__) \
00202 || defined(__AVR_ATtiny22__)
00203
00204     #define SLEEP_MODE_IDLE         0
00205     #define SLEEP_MODE_PWR_DOWN     _BV(SM)
00206
00207     #define set_sleep_mode(mode) \
00208     do { \
00209         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~__BV(SM)) | (mode)); \
00210     } while(0)
00211
00212 #elif defined(__AVR_ATtiny167__) \
00213 || defined(__AVR_ATtiny87__)
00214
00215     #define SLEEP_MODE_IDLE         0
00216     #define SLEEP_MODE_ADC          _BV(SM0)
00217     #define SLEEP_MODE_PWR_DOWN     _BV(SM1)
00218
00219     #define set_sleep_mode(mode) \
00220     do { \
00221         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~(_BV(SM0) | _BV(SM1))) | (mode)); \
00222     } while(0)
00223
00224 #elif defined(__AVR_AT90S4434__) \
00225 || defined(__AVR_AT76C711__) \
00226 || defined(__AVR_AT90S8535__) \
00227 || defined(__AVR_ATmega103__) \
00228 || defined(__AVR_ATmega161__) \
00229 || defined(__AVR_ATmega163__) \
00230 || defined(__AVR_ATmega16HVB__) \
00231 || defined(__AVR_ATmega16HVBREVB__) \
00232 || defined(__AVR_ATmega32HVB__) \
00233 || defined(__AVR_ATmega32HVBREVB__) \
00234 || defined(__AVR_ATtiny13__) \
00235 || defined(__AVR_ATtiny13A__) \
00236 || defined(__AVR_ATtiny15__) \
00237 || defined(__AVR_ATtiny24__) \
00238 || defined(__AVR_ATtiny24A__) \
00239 || defined(__AVR_ATtiny44__) \
00240 || defined(__AVR_ATtiny44A__) \
00241 || defined(__AVR_ATtiny84__) \
00242 || defined(__AVR_ATtiny84A__) \
00243 || defined(__AVR_ATtiny25__) \
00244 || defined(__AVR_ATtiny45__) \
00245 || defined(__AVR_ATtiny48__) \
00246 || defined(__AVR_ATtiny85__) \
00247 || defined(__AVR_ATtiny88__)
00248
00249     #define SLEEP_MODE_IDLE         0
00250     #define SLEEP_MODE_ADC          _BV(SM0)
00251     #define SLEEP_MODE_PWR_DOWN     _BV(SM1)
00252     #define SLEEP_MODE_PWR_SAVE     (_BV(SM0) | _BV(SM1))
00253
00254     #define set_sleep_mode(mode) \
```

```
00255     do { \
00256         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~(_BV(SM0) | _BV(SM1))) | (mode)); \
00257     } while(0)
00258 
00259 #elif defined(__AVR_ATtiny2313__) \
00260 || defined(__AVR_ATtiny2313A__) \
00261 || defined(__AVR_ATtiny4313__)
00262 
00263     #define SLEEP_MODE_IDLE         0
00264     #define SLEEP_MODE_PWR_DOWN     (_BV(SM0) | _BV(SM1))
00265     #define SLEEP_MODE_STANDBY      _BV(SM1)
00266 
00267     #define set_sleep_mode(mode) \
00268     do { \
00269         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~(_BV(SM0) | _BV(SM1))) | (mode)); \
00270     } while(0)
00271 
00272 #elif defined(__AVR_AT94K__)
00273 
00274     #define SLEEP_MODE_IDLE         0
00275     #define SLEEP_MODE_PWR_DOWN     _BV(SM1)
00276     #define SLEEP_MODE_PWR_SAVE     (_BV(SM0) | _BV(SM1))
00277 
00278     #define set_sleep_mode(mode) \
00279     do { \
00280         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~(_BV(SM0) | _BV(SM1))) | (mode)); \
00281     } while(0)
00282 
00283 #elif defined(__AVR_ATtiny26__) \
00284 || defined(__AVR_ATtiny261__) \
00285 || defined(__AVR_ATtiny261A__) \
00286 || defined(__AVR_ATtiny461__) \
00287 || defined(__AVR_ATtiny461A__) \
00288 || defined(__AVR_ATtiny861__) \
00289 || defined(__AVR_ATtiny861A__) \
00290 || defined(__AVR_ATtiny43U__)
00291 
00292     #define SLEEP_MODE_IDLE         0
00293     #define SLEEP_MODE_ADC          _BV(SM0)
00294     #define SLEEP_MODE_PWR_DOWN     _BV(SM1)
00295     #define SLEEP_MODE_STANDBY      (_BV(SM0) | _BV(SM1))
00296 
00297     #define set_sleep_mode(mode) \
00298     do { \
00299         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~(_BV(SM0) | _BV(SM1))) | (mode)); \
00300     } while(0)
00301 
00302 #elif defined(__AVR_AT90PWM216__) \
00303 || defined(__AVR_AT90PWM316__) \
00304 || defined(__AVR_AT90PWM81__)
00305 
00306     #define SLEEP_MODE_IDLE         0
00307     #define SLEEP_MODE_ADC          _BV(SM0)
00308     #define SLEEP_MODE_PWR_DOWN     _BV(SM1)
00309     #define SLEEP_MODE_STANDBY      (_BV(SM1) | _BV(SM2))
00310 
00311     #define set_sleep_mode(mode) \
00312     do { \
00313         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~(_BV(SM0) | _BV(SM1) | _BV(SM2))) | (mode)); \
00314     } while(0)
00315 
00316 #elif defined(__AVR_AT90CAN128__) \
00317 || defined(__AVR_AT90CAN32__) \
00318 || defined(__AVR_AT90CAN64__) \
00319 || defined(__AVR_AT90PWM1__) \
00320 || defined(__AVR_AT90PWM2__) \
00321 || defined(__AVR_AT90PWM2B__) \
00322 || defined(__AVR_AT90PWM3__) \
00323 || defined(__AVR_AT90PWM3B__) \
00324 || defined(__AVR_AT90USB162__) \
00325 || defined(__AVR_AT90USB82__) \
00326 || defined(__AVR_AT90USB1286__) \
00327 || defined(__AVR_AT90USB1287__) \
00328 || defined(__AVR_AT90USB646__) \
00329 || defined(__AVR_AT90USB647__) \
00330 || defined(__AVR_ATmega128__) \
00331 || defined(__AVR_ATmega1280__) \
00332 || defined(__AVR_ATmega1281__) \
00333 || defined(__AVR_ATmega1284P__) \
00334 || defined(__AVR_ATmega128RFA1__) \
00335 || defined(__AVR_ATmega16__) \
00336 || defined(__AVR_ATmega16A__) \
00337 || defined(__AVR_ATmega162__) \
00338 || defined(__AVR_ATmega164A__) \
00339 || defined(__AVR_ATmega164P__) \
00340 || defined(__AVR_ATmega165__) \
00341 || defined(__AVR_ATmega165A__) \
00342 || defined(__AVR_ATmega165P__) \
00343 || defined(__AVR_ATmega168__) \
```

```
00344 || defined(__AVR_ATmega168A__) \
00345 || defined(__AVR_ATmega168P__) \
00346 || defined(__AVR_ATmega169__) \
00347 || defined(__AVR_ATmega169A__) \
00348 || defined(__AVR_ATmega169P__) \
00349 || defined(__AVR_ATmega169PA__) \
00350 || defined(__AVR_ATmega16HVA__) \
00351 || defined(__AVR_ATmega16HVA2__) \
00352 || defined(__AVR_ATmega16M1__) \
00353 || defined(__AVR_ATmega16U2__) \
00354 || defined(__AVR_ATmega16U4__) \
00355 || defined(__AVR_ATmega2560__) \
00356 || defined(__AVR_ATmega2561__) \
00357 || defined(__AVR_ATmega32__) \
00358 || defined(__AVR_ATmega323__) \
00359 || defined(__AVR_ATmega324A__) \
00360 || defined(__AVR_ATmega324P__) \
00361 || defined(__AVR_ATmega324PA__) \
00362 || defined(__AVR_ATmega325__) \
00363 || defined(__AVR_ATmega325A__) \
00364 || defined(__AVR_ATmega3250__) \
00365 || defined(__AVR_ATmega3250A__) \
00366 || defined(__AVR_ATmega328__) \
00367 || defined(__AVR_ATmega328P__) \
00368 || defined(__AVR_ATmega329__) \
00369 || defined(__AVR_ATmega329A__) \
00370 || defined(__AVR_ATmega329P__) \
00371 || defined(__AVR_ATmega329PA__) \
00372 || defined(__AVR_ATmega3290__) \
00373 || defined(__AVR_ATmega3290A__) \
00374 || defined(__AVR_ATmega3290P__) \
00375 || defined(__AVR_ATmega32C1__) \
00376 || defined(__AVR_ATmega32M1__) \
00377 || defined(__AVR_ATmega32U2__) \
00378 || defined(__AVR_ATmega32U4__) \
00379 || defined(__AVR_ATmega32U6__) \
00380 || defined(__AVR_ATmega406__) \
00381 || defined(__AVR_ATmega48__) \
00382 || defined(__AVR_ATmega48A__) \
00383 || defined(__AVR_ATmega48P__) \
00384 || defined(__AVR_ATmega64__) \
00385 || defined(__AVR_ATmega640__) \
00386 || defined(__AVR_ATmega644__) \
00387 || defined(__AVR_ATmega644A__) \
00388 || defined(__AVR_ATmega644P__) \
00389 || defined(__AVR_ATmega644PA__) \
00390 || defined(__AVR_ATmega645__) \
00391 || defined(__AVR_ATmega645A__) \
00392 || defined(__AVR_ATmega645P__) \
00393 || defined(__AVR_ATmega6450__) \
00394 || defined(__AVR_ATmega6450A__) \
00395 || defined(__AVR_ATmega6450P__) \
00396 || defined(__AVR_ATmega649__) \
00397 || defined(__AVR_ATmega649A__) \
00398 || defined(__AVR_ATmega6490__) \
00399 || defined(__AVR_ATmega6490A__) \
00400 || defined(__AVR_ATmega6490P__) \
00401 || defined(__AVR_ATmega649P__) \
00402 || defined(__AVR_ATmega64C1__) \
00403 || defined(__AVR_ATmega64HVE__) \
00404 || defined(__AVR_ATmega64M1__) \
00405 || defined(__AVR_ATmega8__) \
00406 || defined(__AVR_ATmega8515__) \
00407 || defined(__AVR_ATmega8535__) \
00408 || defined(__AVR_ATmega88__) \
00409 || defined(__AVR_ATmega88A__) \
00410 || defined(__AVR_ATmega88P__) \
00411 || defined(__AVR_ATmega88PA__) \
00412 || defined(__AVR_ATmega8HVA__) \
00413 || defined(__AVR_ATmega8U2__)
00414
00415
00416     #define SLEEP_MODE_IDLE         (0)
00417     #define SLEEP_MODE_ADC          _BV(SM0)
00418     #define SLEEP_MODE_PWR_DOWN     _BV(SM1)
00419     #define SLEEP_MODE_PWR_SAVE     (_BV(SM0) | _BV(SM1))
00420     #define SLEEP_MODE_STANDBY      (_BV(SM1) | _BV(SM2))
00421     #define SLEEP_MODE_EXT_STANDBY  (_BV(SM0) | _BV(SM1) | _BV(SM2))
00422
00423
00424     #define set_sleep_mode(mode) \
00425     do { \
00426         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~(_BV(SM0) | _BV(SM1) | _BV(SM2))) | (mode)); \
00427     } while(0)
00428
00429 #elif defined(__AVR_ATxmega16A4__) \
00430 || defined(__AVR_ATxmega16D4__) \
00431 || defined(__AVR_ATxmega32A4__) \
00432 || defined(__AVR_ATxmega32D4__) \
```

```
00433 || defined(__AVR_ATxmega64A1__) \
00434 || defined(__AVR_ATxmega64A1U__) \
00435 || defined(__AVR_ATxmega64A3__) \
00436 || defined(__AVR_ATxmega64D3__) \
00437 || defined(__AVR_ATxmega128A1__) \
00438 || defined(__AVR_ATxmega128A1U__) \
00439 || defined(__AVR_ATxmega128A3__) \
00440 || defined(__AVR_ATxmega128D3__) \
00441 || defined(__AVR_ATxmega192A3__) \
00442 || defined(__AVR_ATxmega192D3__) \
00443 || defined(__AVR_ATxmega256A3__) \
00444 || defined(__AVR_ATxmega256D3__) \
00445 || defined(__AVR_ATxmega256A3B__)
00446
00447     #define SLEEP_MODE_IDLE         (0)
00448     #define SLEEP_MODE_PWR_DOWN     (SLEEP_SMODE1_bm)
00449     #define SLEEP_MODE_PWR_SAVE     (SLEEP_SMODE1_bm | SLEEP_SMODE0_bm)
00450     #define SLEEP_MODE_STANDBY      (SLEEP_SMODE2_bm | SLEEP_SMODE1_bm)
00451     #define SLEEP_MODE_EXT_STANDBY  (SLEEP_SMODE2_bm | SLEEP_SMODE1_bm | SLEEP_SMODE0_bm)
00452
00453     #define set_sleep_mode(mode) \
00454     do { \
00455         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~(SLEEP_SMODE2_bm | SLEEP_SMODE1_bm | SLEEP_SMODE0_bm)) | (mode)); \
00456     } while(0)
00457
00458 #elif defined(__AVR_AT90SCR100__)
00459
00460     #define SLEEP_MODE_IDLE         (0)
00461     #define SLEEP_MODE_PWR_DOWN     _BV(SM1)
00462     #define SLEEP_MODE_PWR_SAVE     (_BV(SM0) | _BV(SM1))
00463     #define SLEEP_MODE_STANDBY      (_BV(SM1) | _BV(SM2))
00464     #define SLEEP_MODE_EXT_STANDBY  (_BV(SM0) | _BV(SM1) | _BV(SM2))
00465
00466     #define set_sleep_mode(mode) \
00467     do { \
00468         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~(_BV(SM0) | _BV(SM1) | _BV(SM2))) | (mode)); \
00469     } while(0)
00470
00471 #elif defined(__AVR_ATA6289__)
00472
00473     #define SLEEP_MODE_IDLE                     (0)
00474     #define SLEEP_MODE_SENSOR_NOISE_REDUCTION   (_BV(SM0))
00475     #define SLEEP_MODE_PWR_DOWN                 (_BV(SM1))
00476
00477     #define set_sleep_mode(mode) \
00478     do { \
00479         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~(_BV(SM0) | _BV(SM1) | _BV(SM2))) | (mode)); \
00480     } while(0)
00481
00482 #elif defined(__AVR_ATtiny4__) \
00483 || defined(__AVR_ATtiny5__) \
00484 || defined(__AVR_ATtiny9__) \
00485 || defined(__AVR_ATtiny10__) \
00486 || defined(__AVR_ATtiny20__) \
00487 || defined(__AVR_ATtiny40__)
00488
00489     #define SLEEP_MODE_IDLE         0
00490     #define SLEEP_MODE_ADC          _BV(SM0)
00491     #define SLEEP_MODE_PWR_DOWN     _BV(SM1)
00492     #define SLEEP_MODE_STANDBY      _BV(SM2)
00493
00494     #define set_sleep_mode(mode) \
00495     do { \
00496         _SLEEP_CONTROL_REG = ((_SLEEP_CONTROL_REG & ~(_BV(SM0) | _BV(SM1) | _BV(SM2))) | (mode)); \
00497     } while(0)
00498
00499 #else
00500
00501     #error "No SLEEP mode defined for this device."
00502
00503 #endif
00504
00505
00506
00507 /** \ingroup avr_sleep
00508
00509     Put the device in sleep mode. How the device is brought out of sleep mode
00510     depends on the specific mode selected with the set_sleep_mode() function.
00511     See the data sheet for your device for more details. */
00512
00513
00514 #if defined(__DOXYGEN__)
00515
00516 /** \ingroup avr_sleep
00517
00518     Set the SE (sleep enable) bit.
00519 */
00520 extern void sleep_enable (void);
00521
```

```
00522 #else
00523
00524 #define sleep_enable()             \
00525 do {                               \
00526   _SLEEP_CONTROL_REG |= (uint8_t)_SLEEP_ENABLE_MASK;   \
00527 } while(0)
00528
00529 #endif
00530
00531
00532 #if defined(__DOXYGEN__)
00533
00534 /** \ingroup avr_sleep
00535
00536     Clear the SE (sleep enable) bit.
00537 */
00538 extern void sleep_disable (void);
00539
00540 #else
00541
00542 #define sleep_disable()            \
00543 do {                               \
00544   _SLEEP_CONTROL_REG &= (uint8_t)(~_SLEEP_ENABLE_MASK);  \
00545 } while(0)
00546
00547 #endif
00548
00549
00550 /** \ingroup avr_sleep
00551
00552     Put the device into sleep mode.  The SE bit must be set
00553     beforehand, and it is recommended to clear it afterwards.
00554 */
00555 #if defined(__DOXYGEN__)
00556
00557 extern void sleep_cpu (void);
00558
00559 #else
00560
00561 #define sleep_cpu()                                \
00562 do {                                               \
00563   __asm__ __volatile__ ( "sleep" "\n\t" :: );    \
00564 } while(0)
00565
00566 #endif
00567
00568
00569 #if defined(__DOXYGEN__)
00570
00571 extern void sleep_mode (void);
00572
00573 #else
00574
00575 #define sleep_mode() \
00576 do {                 \
00577     sleep_enable();  \
00578     sleep_cpu();     \
00579     sleep_disable(); \
00580 } while (0)
00581
00582 #endif
00583
00584
00585 #if defined(__DOXYGEN__)
00586
00587 extern void sleep_bod_disable (void);
00588
00589 #else
00590
00591 #if defined(BODS) && defined(BODSE)
00592
00593 #define sleep_bod_disable() \
00594 do { \
00595   uint8_t tempreg; \
00596   __asm__ __volatile__("in %[tempreg], %[mcucr]" "\n\t" \
00597                        "ori %[tempreg], %[bods_bodse]" "\n\t" \
00598                        "out %[mcucr], %[tempreg]" "\n\t" \
00599                        "andi %[tempreg], %[not_bodse]" "\n\t" \
00600                        "out %[mcucr], %[tempreg]" \
00601                        : [tempreg] "=&d" (tempreg) \
00602                        : [mcucr] "I" _SFR_IO_ADDR(MCUCR), \
00603                          [bods_bodse] "i" (_BV(BODS) | _BV(BODSE)), \
00604                          [not_bodse] "i" (~_BV(BODSE))); \
00605 } while (0)
00606
00607 #endif
00608
00609 #endif
00610
```

```
00611
00612 /*@}*/
00613
00614 #endif /* _AVR_SLEEP_H_ */
```