```
/* Copyright (c) 2002,2007 Michael Stumpf
   All rights reserved.

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

   * Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.

   * 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.

   * Neither the name of the copyright holders nor the names of
     contributors may be used to endorse or promote products derived
     from this software without specific prior written permission.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
  ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS 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. */

/* $Id$ */

/*
   ctype.h - character conversion macros and ctype macros

  Author : Michael Stumpf
           Michael.Stumpf@t-online.de
*/

#ifndef __CTYPE_H_
#define __CTYPE_H_ 1

#ifndef __ATTR_CONST__
#define __ATTR_CONST__ __attribute__((__const__))
#endif

#ifdef __cplusplus
extern "C" {
#endif

/** \file */
/** \defgroup ctype <ctype.h>: Character Operations
    These functions perform various operations on characters.

    \code #include <ctype.h>\endcode

*/

/** \name Character classification routines

    These functions perform character classification. They return true or
    false status depending whether the character passed to the function falls
    into the function's classification (i.e. isdigit() returns true if its
    argument is any value '0' though '9', inclusive). If the input is not
    an unsigned char value, all of this function return false.*/

 /* @{ */
```

```
/** \ingroup ctype

    Checks for an alphanumeric character. It is equivalent to <tt>(isalpha(c)
    || isdigit(c))</tt>. */

extern int isalnum(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Checks for an alphabetic character. It is equivalent to <tt>(isupper(c) ||
    islower(c))</tt>. */

extern int isalpha(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Checks whether \c c is a 7-bit unsigned char value that fits into the
    ASCII character set. */

extern int isascii(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Checks for a blank character, that is, a space or a tab. */

extern int isblank(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Checks for a control character. */

extern int iscntrl(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Checks for a digit (0 through 9). */

extern int isdigit(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Checks for any printable character except space. */

extern int isgraph(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Checks for a lower-case character. */

extern int islower(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Checks for any printable character including space. */

extern int isprint(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Checks for any printable character which is not a space or an alphanumeric
    character. */

extern int ispunct(int __c) __ATTR_CONST__;

/** \ingroup ctype
```

```
    Checks for white-space characters.  For the avr-libc library, these are:
    space, form-feed ('\\f'), newline ('\\n'), carriage return ('\\r'),
    horizontal tab ('\\t'), and vertical tab ('\\v'). */

extern int isspace(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Checks for an uppercase letter. */

extern int isupper(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Checks for a hexadecimal digits, i.e. one of 0 1 2 3 4 5 6 7 8 9 a b c d e
    f A B C D E F. */

extern int isxdigit(int __c) __ATTR_CONST__;

/* @} */

/** \name Character convertion routines

    This realization permits all possible values of integer argument.
    The toascii() function clears all highest bits. The tolower() and
    toupper() functions return an input argument as is, if it is not an
    unsigned char value. */

/* @{ */

/** \ingroup ctype

    Converts \c c to a 7-bit unsigned char value that fits into the ASCII
    character set, by clearing the high-order bits.

    \warning Many people will be unhappy if you use this function. This
    function will convert accented letters into random characters. */

extern int toascii(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Converts the letter \c c to lower case, if possible. */

extern int tolower(int __c) __ATTR_CONST__;

/** \ingroup ctype

    Converts the letter \c c to upper case, if possible. */

extern int toupper(int __c) __ATTR_CONST__;

/* @} */

#ifdef __cplusplus
}
#endif

#endif
```