```c
/* Copyright (c) 2002,2007 Marek Michalkiewicz
   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. */

/* $Id$ */

/*
   string.h

   Contributors:
     Created by Marek Michalkiewicz <marekm@linux.org.pl>
 */

#ifndef _STRING_H_
#define _STRING_H_ 1

#define __need_NULL
#define __need_size_t
#include <stddef.h>

#ifndef __ATTR_PURE__
#define __ATTR_PURE__ __attribute__((__pure__))
#endif

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

#ifdef __cplusplus
extern "C" {
#endif

/** \file */
/** \defgroup avr_string <string.h>: Strings
    \code #include <string.h> \endcode

    The string functions perform string operations on NULL terminated
    strings.

    \note If the strings you are working on resident in program space (flash),
    you will need to use the string functions described in \ref avr_pgmspace. */
```

```c
/** \ingroup avr_string

    This macro finds the first (least significant) bit set in the
    input value.

    This macro is very similar to the function ffs() except that
    it evaluates its argument at compile-time, so it should only
    be applied to compile-time constant expressions where it will
    reduce to a constant itself.
    Application of this macro to expressions that are not constant
    at compile-time is not recommended, and might result in a huge
    amount of code generated.

    \returns The _FFS() macro returns the position of the first
    (least significant) bit set in the word val, or 0 if no bits are set.
    The least significant bit is position 1.  Only 16 bits of argument
    are evaluted.
*/
#if defined(__DOXYGEN__)
#define _FFS(x)
#else  /* !DOXYGEN */
#define _FFS(x) \
  (1        \
   + (((x) & 1) == 0)   \
   + (((x) & 3) == 0)   \
   + (((x) & 7) == 0)   \
   + (((x) & 017) == 0)   \
   + (((x) & 037) == 0)   \
   + (((x) & 077) == 0)   \
   + (((x) & 0177) == 0)    \
   + (((x) & 0377) == 0)    \
   + (((x) & 0777) == 0)    \
   + (((x) & 01777) == 0)   \
   + (((x) & 03777) == 0)   \
   + (((x) & 07777) == 0)   \
   + (((x) & 017777) == 0) \
   + (((x) & 037777) == 0) \
   + (((x) & 077777) == 0) \
   - (((x) & 0177777) == 0) * 16)
#endif /* DOXYGEN */

extern int ffs (int __val) __ATTR_CONST__;
extern int ffsl (long __val) __ATTR_CONST__;
extern int ffsll (long long __val) __ATTR_CONST__;
extern void *memccpy(void *, const void *, int, size_t);
extern void *memchr(const void *, int, size_t) __ATTR_PURE__;
extern int memcmp(const void *, const void *, size_t) __ATTR_PURE__;
extern void *memcpy(void *, const void *, size_t);
extern void *memmem(const void *, size_t, const void *, size_t) __ATTR_PURE__;
extern void *memmove(void *, const void *, size_t);
extern void *memrchr(const void *, int, size_t) __ATTR_PURE__;
extern void *memset(void *, int, size_t);
extern char *strcat(char *, const char *);
extern char *strchr(const char *, int) __ATTR_PURE__;
extern char *strchrnul(const char *, int) __ATTR_PURE__;
extern int strcmp(const char *, const char *) __ATTR_PURE__;
extern char *strcpy(char *, const char *);
extern int strcasecmp(const char *, const char *) __ATTR_PURE__;
extern char *strcasestr(const char *, const char *) __ATTR_PURE__;
extern size_t strcspn(const char *__s, const char *__reject) __ATTR_PURE__;
extern char *strdup(const char *s1);
extern size_t strlcat(char *, const char *, size_t);
extern size_t strlcpy(char *, const char *, size_t);
extern size_t strlen(const char *) __ATTR_PURE__;
extern char *strlwr(char *);
extern char *strncat(char *, const char *, size_t);
extern int strncmp(const char *, const char *, size_t) __ATTR_PURE__;
extern char *strncpy(char *, const char *, size_t);
extern int strncasecmp(const char *, const char *, size_t) __ATTR_PURE__;
extern size_t strnlen(const char *, size_t) __ATTR_PURE__;
extern char *strpbrk(const char *__s, const char *__accept) __ATTR_PURE__;
extern char *strrchr(const char *, int) __ATTR_PURE__;
extern char *strrev(char *);
extern char *strsep(char **, const char *);
extern size_t strspn(const char *__s, const char *__accept) __ATTR_PURE__;
extern char *strstr(const char *, const char *) __ATTR_PURE__;
extern char *strtok(char *, const char *);
extern char *strtok_r(char *, const char *, char **);
extern char *strupr(char *);

#ifdef __cplusplus
}
#endif

#endif /* _STRING_H_ */
```