/* ———————————————————————————————— *

src2tex* version 2.12 (Sep 9, 1996)

Kazuo Amano† and Shinichi Nomoto‡

* ———————————————————————————————— */

/* ———————————— modflag.c ———————————— */

```
#include <stdio.h>
#include "src2tex.h"

extern int BAS_flag;
extern int C_flag;
extern int CBL_flag;
extern int F77_flag;
extern int HTML_flag;
extern int JAVA_flag;
extern int LISP_flag;
extern int MAKE_flag;
extern int PAS_flag;
extern int PERL_flag;
extern int SH_flag;
extern int TCL_flag;

extern int MAC_flag;
extern int MAP_flag;
extern int MAT_flag;
extern int MLAB_flag;
extern int MPAD_flag;
extern int RED_flag;

extern int *dec_buf_ptr();
extern int *inc_buf_ptr();
extern void set_lang_flag();
```

/* ———————————— get comment flag ———————————— */

/* When `stat_flag`= 0, the function `get_comment_flag()` increases `stat_flag` first and determines language type by using functions

† Department of Mathematics, Josai University, Keyakidai, Sakado, Saitama, 350-02, JAPAN (kamano@po.iijnet.or.jp)

‡ Department of Mathematics, Josai University, Keyakidai, Sakado, Saitama, 350-02, JAPAN (m93093@crescent.josai.ac.jp)

`set_bas_flag()`, `set_c_flag()`, `set_f77_flag()`, `set_lisp_flag()`, `set_pas_flag()`, etc.

For instance, if input file is written in C, then language flags `BAS_flag`, `C_flag`, `F77_flag`, `LISP_flag`, `PAS_flag`, *etc* are set as follows:

$$\texttt{BAS\_flag} = 0, \texttt{C\_flag} = 1, \texttt{F77\_flag} = 0, \texttt{LISP\_flag} = 0, \texttt{PAS\_flag} = 0\ , \cdots .$$

In the main part of function `get_comment_flag()`, the procedure is branched into five cases. In each case, comment flag `com_flag` is set to 1, *i.e.*, `com_flag`= 1, if and only if integer pointer `buf_ptr` points a character of comment area of source program.

$$\texttt{com\_flag} = \begin{cases} 1 & \text{if } \texttt{buf\_ptr} \text{ points a character of comment area} \\ 0 & \text{otherwise .} \end{cases}$$

As is easily seen, it is not so difficult to distinguish comment area from program area, if language type is determined in advance. */

```
int get_comment_flag(buf_ptr)
int *buf_ptr;
{
    static int stat_flag = 0;           /* status flag                    */
    static int paren_counter = 0;       /* parentheses counter            */
    static int brace_counter = 0;       /* brace counter                  */
    static int com_flag = 0;            /* comment flag                   */
    static int char_counter = 0;        /* character counter              */
    static int qt_counter = 0;          /* single quotation counter       */
    static int bqt_counter = 0;         /* back quotation counter         */
    static int dqt_counter = 0;         /* double quotation counter       */
    static int slash_counter = 0;       /* slash counter                  */
    static int rgx_counter = 0;         /* regular expression counter     */
    static int warn_flag = 0;           /* warning flag                   */
    int *ptr;
    char c_2, c_1, c0, c1, c2, c3, c4, c5, c6;

    /* determine language                                                 */
    if (stat_flag == 0)
      {
         ++stat_flag;
         set_lang_flag(buf_ptr);
      }
    /* get a neighborhood of buf_ptr                                      */
    ptr = dec_buf_ptr(buf_ptr);
    c_1 = *ptr;
    ptr = dec_buf_ptr(ptr);
    c_2 = *ptr;
    c0 = *buf_ptr;
    ptr = inc_buf_ptr(buf_ptr);
    c1 = *ptr;
    ptr = inc_buf_ptr(ptr);
    c2 = *ptr;
    ptr = inc_buf_ptr(ptr);
    c3 = *ptr;
    ptr = inc_buf_ptr(ptr);
    c4 = *ptr;
    ptr = inc_buf_ptr(ptr);
    c5 = *ptr;
    ptr = inc_buf_ptr(ptr);
    c6 = *ptr;
```

```
/* set comment flag of BASIC                                          */
if (BAS_flag == 1)
  {
    if ((c0 == '"') && (com_flag == 0))
      {
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if (dqt_counter == 0)
      {
        if ((com_flag == 1) && (c_1 == '\n'))
          com_flag = 0;
        if (char_counter == 0)
          {
            if ((com_flag == 0) && (c_1 <= ' ') && (c0 == 0x27))
              {
                com_flag = 1;
                if ((c1 == '{') && (c2 == '\\') & (c3 == ' '))
                  *buf_ptr = ' ';
              }
            if ((com_flag == 0) && (c_1 <= ' ')
                && ((c0 == 'R') || (c0 == 'r'))
                && ((c1 == 'E') || (c1 == 'e'))
                && ((c2 == 'M') || (c2 == 'm')))
              com_flag = 1;
          }
      }
    if (c0 == '\n')
      char_counter = 0;
    else
      {
        if (char_counter != 0)
          ++char_counter;
        else
          if ((c0 > '9') || ((c0 > ' ') && (c0 < '0')))
            ++char_counter;
      }
    return com_flag;
  }
/* set comment flag of C or MACSYMA                                   */
if ((C_flag == 1) || (MAC_flag == 1))
  {
    if ((((c_2 == '\\') && (c_1 == '\\')) || (c_1 != '\\'))
        && (c0 == '"')
        && ((c_1 != 0x27) || (c1 != 0x27)) && (com_flag == 0))
      {
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if (dqt_counter == 0)
      {
        if ((com_flag == 1) && (c_2 == '*') && (c_1 == '/'))
          com_flag = 0;
```

```
        if ((com_flag == 3) && ( c_1 == '\n'))
          com_flag = 0;
        if ((com_flag == 0) && (c0 == '/') && (c1 == '*'))
          com_flag = 1;
        if ((com_flag == 0) && (c0 == '/') && (c1 == '/'))
          com_flag = 3;
      }
    return (com_flag % 2);
  }
/* set comment flag of COBOL                                          */
if (CBL_flag == 1)
  {
    if ((com_flag == 1) && (char_counter == 0)
        && (c_1 == '\n'))
      com_flag = 0;
    if ((com_flag == 0) && (char_counter == 0)
        && ((c6 == '*') || (c6 == '/')))
      com_flag = 1;
    if (c0 == '\n')
      char_counter = 0;
    else
      ++char_counter;
    return com_flag;
  }
/* set comment flag of FORTRAN                                        */
if (F77_flag == 1)
  {
    if ((c0 == 0x27) && (com_flag == 0))
      {
        ++qt_counter;
        qt_counter %= 2;
      }
    if (qt_counter == 0)
      {
        if((com_flag == 1) && ((c_1 == '\r') || (c_1 == '\n')))
          com_flag = 0;
        if ((com_flag == 1)
            && ((c_1 == 'C') || (c_1 == 'c') || (c_1 == '*'))
            && ((c0 == '\r') || (c0 == '\n')))
          com_flag = 0;
        if (char_counter == 0)
          {
            if ((com_flag == 0) && ((c0 == 'C') || (c0 == 'c')))
              com_flag = 1;
            if ((com_flag == 0) && (c0 == '*'))
              {
                com_flag = 1;
                if ((c1 == '{') && (c2 == '\\') & (c3 == ' '))
                  *buf_ptr = ' ';
              }
          }
      }
    if (c0 == '\n')
```

```
      char_counter = 0;
    else
      ++char_counter;
    return com_flag;
  }
/* set comment flag of HTML                                                      */
if (HTML_flag == 1)
  {
    if ((((c_2 == '\\') && (c_1 == '\\')) || (c_1 != '\\'))
        && (c0 == '"')
        && ((c_1 != 0x27) || (c1 != 0x27)) && (com_flag == 0))
      {
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if (dqt_counter == 0)
      {
        if ((com_flag == 1) && (c_2 == '-') && (c_1 == '>'))
          com_flag = 0;
        if ((com_flag == 0)
            && (c0 == '<') && (c1 == '!') && (c2 = '-'))
          com_flag = 1;
      }
    return com_flag;
  }
/* set comment flag of JAVA                                                      */
if (JAVA_flag == 1)
  {
    if ((((c_2 == '\\') && (c_1 == '\\')) || (c_1 != '\\'))
        && (c0 == '"')
        && ((c_1 != 0x27) || (c1 != 0x27)) && (com_flag == 0))
      {
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if (dqt_counter == 0)
      {
        if ((com_flag == 1) && (c_2 == '*') && (c_1 == '/'))
          com_flag = 0;
        if ((com_flag == 3) && ( c_1 == '\n'))
          com_flag = 0;
        if ((com_flag == 0) && (c0 == '/') && (c1 == '*'))
          com_flag = 1;
        if ((com_flag == 0) && (c0 == '/') && (c1 == '/'))
          com_flag = 3;
      }
    return (com_flag % 2);
  }
/* set comment flag of LISP                                                      */
if (LISP_flag == 1)
  {
    if ((c0 == '"') && (com_flag == 0))
      {
```

```
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if (dqt_counter == 0)
      {
        if ((com_flag == 1) && ( c_1 == '\n'))
          com_flag = 0;
        if ((com_flag == 0) && (c0 == ';'))
          {
            com_flag = 1;
            if ((c1 == '{') && (c2 == '\\') & (c3 == ' '))
              *buf_ptr = ' ';
          }
      }
    return com_flag;
  }
/* set comment flag of MAKE                                                    */
if (MAKE_flag == 1)
  {
    if ((c0 == 0x27) && (com_flag == 0)
        && (bqt_counter == 0) && (dqt_counter == 0))
      {
        ++qt_counter;
        qt_counter %= 2;
      }
    if ((c0 == 0x60) && (com_flag == 0)
        && (qt_counter == 0) && (dqt_counter == 0))
      {
        ++bqt_counter;
        bqt_counter %= 2;
      }
    if ((c0 == '"') && (com_flag == 0)
        && (qt_counter == 0) && (bqt_counter == 0))
      {
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if ((qt_counter == 0) && (bqt_counter == 0) && (dqt_counter == 0))
      {
        if ((com_flag == 1)
            && (c_2 != '\\') && ((c_1 == '\r') || (c_1 == '\n')))
          com_flag = 0;
        if ((com_flag == 1)
            && (c_1 == '#') && ((c0 == '\r') || (c0 == '\n')))
          com_flag = 0;
        if ((com_flag == 0)
            && (c0 == '#'))
          {
            com_flag = 1;
            if ((c1 == '{') && (c2 == '\\') & (c3 == ' '))
              *buf_ptr = ' ';
          }
      }
```

```
    return com_flag;
  }
/* set comment flag of MuPAD                                    */
if (MPAD_flag == 1)
  {
    if ((c0 == 0x27) && (com_flag == 0)
        && (bqt_counter == 0) && (dqt_counter == 0))
      {
        ++qt_counter;
        qt_counter %= 2;
      }
    if ((c0 == 0x60) && (com_flag == 0)
        && (qt_counter == 0) && (dqt_counter == 0))
      {
        ++bqt_counter;
        bqt_counter %= 2;
      }
    if ((c0 == '"') && (com_flag == 0)
        && (qt_counter == 0) && (bqt_counter == 0))
      {
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if ((qt_counter == 0) && (bqt_counter == 0) && (dqt_counter == 0))
      {
        if ((com_flag == 1)
            && (c_1 != '\\') && (c0 == '#'))
          com_flag = 0;
        if ((com_flag == 0)
            && ((c_1 <= 0x20) || (c_1 == ';')) && (c0 == '#'))
          com_flag = 1;
      }
    return com_flag;
  }
/* set comment flag of PASCAL                                   */
if (PAS_flag == 1)
  {
    if ((c0 == 0x27) && (com_flag == 0))
      {
        ++qt_counter;
        qt_counter %= 2;
      }
      if ((qt_counter == 0) && (c_1 != '\\') && (c0 == '{'))
        ++brace_counter;
      if ((qt_counter == 0) && (c_1 != '\\') && (c0 == '}'))
        --brace_counter;
    if (qt_counter == 0)
      {
        if ((brace_counter == 0) && (com_flag == 1) && (c_1 == '}'))
          com_flag = 0;
        if ((brace_counter == 1) && (com_flag == 0) && (c0 == '{'))
          com_flag = 1;
        if ((com_flag == 3) && (c_2 == '*') && (c_1 == ')'))
```

```
        com_flag = 0;
      if ((com_flag == 0) && (c0 == '(') && (c1 == '*'))
        com_flag = 3;
    }
  return (com_flag % 2);
}
```

/* set comment flag of PERL */

```
if (PERL_flag == 1)
  {
    if ((((c_2 != '$') && (c_1 == 'm') && (c0 == '!'))
         || ((c_1 != '\\') && (c0 == '?')))
        && (com_flag == 0) && (rgx_counter == 0)
        && (paren_counter == 0) && (brace_counter == 0)
        && (qt_counter == 0) && (bqt_counter == 0)
        && (dqt_counter == 0) && (slash_counter == 0))
        ++rgx_counter;
    if ((c_1 != '\\') && ((c0 == '!') || (c0 == '?'))
        && (com_flag == 0) && (rgx_counter == 1)
        && (paren_counter == 0) && (brace_counter == 0)
        && (qt_counter == 0) && (bqt_counter == 0)
        && (dqt_counter == 0) && (slash_counter == 0))
        --rgx_counter;
    if ((c_1 != '\\') && ((c0 == '(') || (c0 == ')'))
        && (com_flag == 0) && (rgx_counter == 0)
        && (brace_counter == 0)
        && (qt_counter == 0) && (dqt_counter == 0)
        && (bqt_counter == 0) && (slash_counter == 0))
      {
        ++paren_counter;
        paren_counter %= 2;
      }
    if ((c_1 != '\\') && (c0 == '{')
        && (((c_1 >= 'A') && (c_1 <= 'Z'))
            || ((c_1 >= 'a') && (c_1 <= 'z')))
        && (com_flag == 0) && (rgx_counter == 0)
        && (paren_counter == 0) && (brace_counter == 0)
        && (qt_counter == 0) && (bqt_counter == 0)
        && (dqt_counter == 0) && (slash_counter == 0))
      ++brace_counter;
    if ((c_1 != '\\') && (c0 == '}')
        && (com_flag == 0) && (rgx_counter == 0)
        && (paren_counter == 0) && (brace_counter == 1)
        && (qt_counter == 0) && (bqt_counter == 0)
        && (dqt_counter == 0) && (slash_counter == 0))
      --brace_counter;
    if ((c_1 != '\\') && (c0 == 0x27)
        && (com_flag == 0) && (rgx_counter == 0)
        && (paren_counter == 0) && (brace_counter == 0)
        && (bqt_counter == 0)
        && (dqt_counter == 0) && (slash_counter == 0))
      {
        ++qt_counter;
        qt_counter %= 2;
```

```
  }
if ( (((c_1 == 'q') && (c0 == '|'))
      || ((c_1 != '\\') && (c0 == '|')))
    && (com_flag == 0) && (rgx_counter == 0)
    && (paren_counter == 0) && (brace_counter == 0)
    && (bqt_counter == 0)
    && (dqt_counter == 0) && (slash_counter == 0))
  {
    qt_counter += 2;
    qt_counter %= 4;
  }
if ((c_1 != '\\') && (c0 == 0x60)
    && (com_flag == 0) && (rgx_counter == 0)
    && (paren_counter == 0) && (brace_counter == 0)
    && (qt_counter == 0) && (bqt_counter == 0)
    && (dqt_counter == 0) && (slash_counter == 0))
  bqt_counter = 1;
if ((c_1 != '\\') && (c0 == 0x60)
    && (com_flag == 0) && (rgx_counter == 0)
    && (paren_counter == 0) && (brace_counter == 0)
    && (qt_counter == 0) && (bqt_counter == 1)
    && (dqt_counter == 0) && (slash_counter == 0))
  bqt_counter = 0;
if ((c_1 != '\\') && (c0 == '"')
    && (com_flag == 0) && (rgx_counter == 0)
    && (paren_counter == 0) && (brace_counter == 0)
    && (qt_counter == 0) && (bqt_counter == 0)
    && (slash_counter == 0))
  {
    ++dqt_counter;
    dqt_counter %= 2;
  }
if ((c_1 == ' ') && (c0 == '/')
    && (com_flag == 0) && (rgx_counter == 0)
    && (paren_counter == 0) && (brace_counter == 0)
    && (qt_counter == 0) && (bqt_counter == 0)
    && (dqt_counter == 0) && (slash_counter == 0))
  slash_counter = 2;
if ((c_2 != '$') && ((c_1 >= 'a') && (c_1 <= 'z')) && (c0 == '/')
    && (com_flag == 0) && (rgx_counter == 0)
    && (paren_counter == 0) && (brace_counter == 0)
    && (qt_counter == 0) && (bqt_counter == 0)
    && (dqt_counter == 0) && (slash_counter == 0))
  slash_counter = 3;
if ((((c_1 != '\\') || ((c_2 == '\\') && (c_1 == '\\')))
     && (c0 == '/'))
    && (com_flag == 0) && (rgx_counter == 0)
    && (paren_counter == 0) && (brace_counter == 0)
    && (qt_counter == 0) && (bqt_counter == 0)
    && (dqt_counter == 0) && (slash_counter != 0))
  --slash_counter;
if ((rgx_counter == 0)
    && (paren_counter == 0) && (brace_counter == 0)
```

```
          && (qt_counter == 0) && (bqt_counter == 0)
          && (dqt_counter == 0) && (slash_counter == 0))
        {
/*
          if ((com_flag == 1)
              && (c_2 != '\\') && ((c_1 == '\r') || (c_1 == '\n')))
            com_flag = 0;
*/
          if ((com_flag == 1)
              && ((c_1 == '\r') || (c_1 == '\n')))
            com_flag = 0;
          if ((com_flag == 1)
              && (c_1 == '#') && ((c0 == '\r') || (c0 == '\n')))
            com_flag = 0;
          if ((com_flag == 0)
              && (rgx_counter == 0)
              && (paren_counter == 0) && (brace_counter == 0)
              && (qt_counter == 0) && (bqt_counter == 0)
              && (dqt_counter == 0) && (slash_counter == 0)
              && ((stat_flag++ == 1)
                  || (c_1 == '\t') || (c_1 == '\n')
                  || (c_1 == ' ') || (c_1 == ';'))
              && (c0 == '#'))
            {
              com_flag = 1;
              if ((c1 == '{') && (c2 == '\\') & (c3 == ' '))
                *buf_ptr = ' ';
            }
        }
      if (c0 == '\n')
        char_counter = 0;
      else
        ++char_counter;
      return com_flag;
    }
  /* set comment flag of SHELL                                                    */
  if (SH_flag == 1)
    {
      if ((c0 == 0x27) && (com_flag == 0)
          && (bqt_counter == 0) && (dqt_counter == 0))
        {
          ++qt_counter;
          qt_counter %= 2;
        }
      if ((c0 == 0x60) && (com_flag == 0)
          && (qt_counter == 0) && (dqt_counter == 0))
        {
          ++bqt_counter;
          bqt_counter %= 2;
        }
      if ((c0 == '"') && (com_flag == 0)
          && (qt_counter == 0) && (bqt_counter == 0))
        {
```

```
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if ((qt_counter == 0) && (bqt_counter == 0) && (dqt_counter == 0))
      {
        if ((com_flag == 1)
            && (c_2 != '\\') && ((c_1 == '\r') || (c_1 == '\n')))
          com_flag = 0;
        if ((com_flag == 1)
            && (c_1 == '#') && ((c0 == '\r') || (c0 == '\n')))
          com_flag = 0;
        if ((com_flag == 0)
            && (c_1 != '$') && (c0 == '#'))
          {
            com_flag = 1;
            if ((c1 == '{') && (c2 == '\\') & (c3 == ' '))
              *buf_ptr = ' ';
          }
      }
    return com_flag;
  }
/* set comment flag of TCL/TK                                          */
if (TCL_flag == 1)
  {
    if ((c0 == 0x27) && (com_flag == 0)
        && (bqt_counter == 0) && (dqt_counter == 0))
      {
        ++qt_counter;
        qt_counter %= 2;
      }
    if ((c0 == 0x60) && (com_flag == 0)
        && (qt_counter == 0) && (dqt_counter == 0))
      {
        ++bqt_counter;
        bqt_counter %= 2;
      }
    if ((c0 == '"') && (com_flag == 0)
        && (qt_counter == 0) && (bqt_counter == 0))
      {
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if ((qt_counter == 0) && (bqt_counter == 0) && (dqt_counter == 0))
      {
        if ((com_flag == 1)
            && (c_2 != '\\') && ((c_1 == '\r') || (c_1 == '\n')))
          com_flag = 0;
        if ((com_flag == 1)
            && (c_1 == '#') && ((c0 == '\r') || (c0 == '\n')))
          com_flag = 0;
        if ((com_flag == 0)
            && (c0 == '#'))
          {
```

```
            com_flag = 1;
            if ((c1 == '{') && (c2 == '\\') & (c3 == ' '))
              *buf_ptr = ' ';
          }
      }
    return com_flag;
  }
/* set comment flag of MAPLE                                        */
if (MAP_flag == 1)
  {
    if ((com_flag == 0)
        && (c_1 != 0x27) && (c0 == 0x27) && (c1 != 0x27))
      {
        ++qt_counter;
        qt_counter %= 2;
      }
    if (qt_counter == 0)
      {
        if ((com_flag == 1)
            && (c_2 != '\\') && ((c_1 == '\r') || (c_1 == '\n')))
          com_flag = 0;
        if ((com_flag == 1)
            && (c_1 == '#') && ((c0 == '\r') || (c0 == '\n')))
          com_flag = 0;
        if ((com_flag == 0)
            && (c0 == '#'))
          {
            com_flag = 1;
            if ((c1 == '{') && (c2 == '\\') & (c3 == ' '))
              *buf_ptr = ' ';
          }
      }
    return com_flag;
  }
/* set comment flag of MATHEMATICA                                  */
if (MAT_flag == 1)
  {
    if ((c0 == '"') && (com_flag == 0))
      {
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if (dqt_counter == 0)
      {
        if ((com_flag == 1) && (c_2 == '*') && (c_1 == ')'))
          com_flag = 0;
        if ((com_flag == 0) && (c0 == '(') && (c1 == '*'))
          com_flag = 1;
      }
    return com_flag;
  }
/* set comment flag of MATLAB                                       */
if (MLAB_flag == 1)
```

```
  {
    if ((c0 == 0x27) && (qt_counter == 1))
      qt_counter = 0;
    if (((c_1 < '0') || ((c_1 > '9') && (c_1 < 'A'))
         || ((c_1 > 'Z') && (c_1 < 'a')) || (c_1 > 'z'))
        && (c_1 != '\\') && (c_1 != '.')
        && (c_1 != ')') && (c_1 != ']')
        && (c0 == 0x27) && (com_flag == 0))
      qt_counter = 1;
    if ((c_1 != '\\') && (c0 == '"') && (com_flag == 0))
      {
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if ((qt_counter == 0) && (dqt_counter == 0))
      {
        if ((com_flag == 1) && ( c_1 == '\n'))
          com_flag = 0;
        if ((com_flag == 0) && ((c0 == '#') || (c0 == '%')))
          {
            com_flag = 1;
            if ((c1 == '{') && (c2 == '\\') & (c3 == ' '))
              *buf_ptr = ' ';
          }
      }
    return com_flag;
  }
/* set comment flag of REDUCE                                              */
if (RED_flag == 1)
  {
    if ((c0 == '"') && (com_flag == 0))
      {
        ++dqt_counter;
        dqt_counter %= 2;
      }
    if (dqt_counter == 0)
      {
        if ((com_flag == 1) && ( c_1 == '\n'))
          com_flag = 0;
        if ((com_flag == 0) && (c0 == '%'))
          {
            com_flag = 1;
            if ((c1 == '{') && (c2 == '\\') & (c3 == ' '))
              *buf_ptr = ' ';
          }
        if ((com_flag == 3) && (c_1 == ';'))
          com_flag = 0;
        if ((com_flag == 0)
            && ((c0 == 'C') || (c0 == 'c'))
            && ((c1 == 'O') || (c1 == 'o'))
            && ((c2 == 'M') || (c2 == 'm'))
            && ((c3 == 'M') || (c3 == 'm'))
            && ((c4 == 'E') || (c4 == 'e'))
```

```
              && ((c5 == 'N') || (c5 == 'n'))
              && ((c6 == 'T') || (c6 == 't')))
            com_flag = 3;
        }
      return (com_flag % 2);
    }
  return com_flag;
}
```

/* ——————————————— get TEX mode flag ——————————————— */

/* This function sets math flag `math_flag` and escape flag `esc_flag` as follows:

$$\texttt{math\_flag}\begin{cases}\neq 0 & \text{if } \texttt{buf\_ptr} \text{ points between \$ and \$ \quad (or \textbackslash(and \textbackslash))}\\ \neq 0 & \text{if } \texttt{buf\_ptr} \text{ points between \$\$ and \$\$ \quad (or \textbackslash[and \textbackslash])}\\ = 0 & \text{otherwise ,}\end{cases}$$

$$\texttt{esc\_flag}\begin{cases}\neq 0 & \text{if } \texttt{buf\_ptr} \text{ points between \{\textbackslash\ and \}}\\ = 0 & \text{otherwise .}\end{cases}$$

And it returns

`tex_flag = ((maht_flag != 0) | (esc_flag != 0))`

at the last stage. */

```
int get_tex_flag(buf_ptr)
int *buf_ptr;
{
  static int math_flag = 0;            /* math flag                                  */
  static int ds_flag = 0;              /* display math flag                          */
  static int esc_flag = 0;             /* escape from quasi-TeX flag                 */
  static int rcs_flag = 0;             /* RCS flag                                   */
  static int brace_counter = 0;        /* brace counter                              */
  char a[4], b[10];                    /* mini buffer a[3]...a[0]= b[0]...b[9]       */
  char c0, c1, c2;
  int i, *ptr, *ptr1, *ptr2, tex_flag, tex_counter;

  /* unset math_flag and esc_flag (TEX→quasi-TEX)                                    */
  ptr = buf_ptr;
  a[0] = *ptr;
  ptr = dec_buf_ptr(ptr);
  a[1] = *ptr;
  ptr = dec_buf_ptr(ptr);
  a[2] = *ptr;
  ptr = dec_buf_ptr(ptr);
  a[3] = *ptr;
#ifdef LATEX
  if (rcs_flag != 0)                   /* reset RCS flag                             */
    if ((a[2] != '\\') && (a[2] != '$')
        && (a[1] == '$') && (a[0] != '$'))
      --rcs_flag;
  if (math_flag != 0)                  /* reset LaTeX math flag                      */
    {
      if (ds_flag == 0)
        {
          if ((a[1] == '$') && (a[0] != '$'))
            if ((a[2] != '\\') && (a[2] != '$'))
```

```
            --math_flag;
          if ((a[2] == '\\') && (a[1] == ')'))
            math_flag = 0;
        }
      else
        {
          if ((a[2] == '\\') && (a[1] == ']'))
            {
              math_flag = 0;
              ds_flag = 0;
            }
        }
    }
#else
    if (rcs_flag != 0)                  /* reset RCS flag                                  */
      if ((a[2] != '\\') && (a[2] != '$')
          && (a[1] == '$') && (a[0] != '$'))
        --rcs_flag;
    if (math_flag != 0)                 /* reset TeX math flag                             */
      {
          if ((a[1] == '$') && (a[0] != '$'))
            {
              if (ds_flag == 0)
                {
                  if ((a[2] != '\\') && (a[2] != '$'))
                    --math_flag;
                }
              else
                {
                  if ((a[3] != '\\') && (a[3] != '$') && (a[2] == '$'))
                    --math_flag;
                  if (math_flag == 0)
                    ds_flag = 0;
                }
            }
      }
#endif
    if (esc_flag != 0)
      {
        if ((a[1] != '\\') && (a[0] == '{'))
          ++brace_counter;
        if ((a[1] != '\\') && (a[0] == '}'))
          --brace_counter;
        if (esc_flag == 2)
          {
            if (brace_counter == 0)
              {
                if (a[0] != '}')
                  {
                    fprintf(stderr, "\nError: brace counter error !!\n");
                    exit(EXIT_FAILURE);
                  }
                --esc_flag;
```

```
              *buf_ptr = ' ';          /* replace a TEX-mode-end-brace with a space          */
            }
        }
      else
        --esc_flag;
    }

  /* set math_flag and esc_flag (quasi-TEX→TEX)                                            */
  ptr = buf_ptr;
  b[0] = *ptr;
  ptr = inc_buf_ptr(ptr);
  b[1] = *ptr;
  ptr = inc_buf_ptr(ptr);
  b[2] = *ptr;
  ptr = inc_buf_ptr(ptr);
  b[3] = *ptr;
  ptr = inc_buf_ptr(ptr);
  b[4] = *ptr;
  ptr = inc_buf_ptr(ptr);
  b[5] = *ptr;
  ptr = inc_buf_ptr(ptr);
  b[6] = *ptr;
  ptr = inc_buf_ptr(ptr);
  b[7] = *ptr;
  ptr = inc_buf_ptr(ptr);
  b[8] = *ptr;
  ptr = inc_buf_ptr(ptr);
  b[9] = *ptr;
#ifdef LATEX
  if (math_flag == 0)
    {
      if ((a[1] != '\\') && (a[1] != '$') && (b[0] == '$'))
        {
          if (b[1] != '$')     /* check RCS keywords here                                   */
            {
              if ((b[1] == 'A') && (b[2] == 'u') && (b[3] == 't')
                  && (b[4] == 'h') && (b[5] == 'o') && (b[6] == 'r')
                  && ((b[7] == ':') || (b[7] == '$')))
                rcs_flag = 2;
              if ((b[1] == 'D') && (b[2] == 'a') && (b[3] == 't')
                  && (b[4] == 'e')
                  && ((b[5] == ':') || (b[5] == '$')))
                rcs_flag = 2;
              if ((b[1] == 'H') && (b[2] == 'e') && (b[3] == 'a')
                  && (b[4] == 'd') && (b[5] == 'e') && (b[6] == 'r')
                  && ((b[7] == ':') || (b[7] == '$')))
                rcs_flag = 2;
              if ((b[1] == 'I') && (b[2] == 'd')
                  && ((b[3] == ':') || (b[3] == '$')))
                rcs_flag = 2;
              if ((b[1] == 'L') && (b[2] == 'o') && (b[3] == 'c')
                  && (b[4] == 'k') && (b[5] == 'e') && (b[6] == 'r')
                  && ((b[7] == ':') || (b[7] == '$')))
```

```
          rcs_flag = 2;
        if ((b[1] == 'L') && (b[2] == 'o') && (b[3] == 'g')
            && ((b[4] == ':') || (b[4] == '$')))
          rcs_flag = 2;
        if ((b[1] == 'R') && (b[2] == 'e') && (b[3] == 'v')
            && (b[4] == 'i') && (b[5] == 's') && (b[6] == 'i')
            && (b[7] == 'o') && (b[8] == 'n')
            && ((b[9] == ':') || (b[9] == '$')))
          rcs_flag = 2;
        if ((b[1] == 'R') && (b[2] == 'C') && (b[3] == 'S')
            && (b[4] == 'f') && (b[5] == 'i') && (b[6] == 'l')
            && (b[7] == 'e')
            && ((b[8] == ':') || (b[8] == '$')))
          rcs_flag = 2;
        if ((b[1] == 'S') && (b[2] == 'o') && (b[3] == 'n')
            && (b[4] == 'y') && (b[5] == 'I') && (b[6] == 'd')
            && ((b[7] == ':') || (b[7] == '$')))
          rcs_flag = 2;
        if ((b[1] == 'S') && (b[2] == 'o') && (b[3] == 'u')
            && (b[4] == 'r') && (b[5] == 'c') && (b[6] == 'e')
            && ((b[7] == ':') || (b[7] == '$')))
          rcs_flag = 2;
        if ((b[1] == 'S') && (b[2] == 't') && (b[3] == 'a')
            && (b[4] == 't') && (b[5] == 'e')
            && ((b[6] == ':') || (b[6] == '$')))
          rcs_flag = 2;
      }
    if ((BAS_flag == 0) && (PERL_flag == 0))
      {
        if (b[1] != '$')
          math_flag = 2;
      }
    else                /* else-part takes care of BASIC or PERL string variables     */
      {
        if ((BAS_flag != 0)
            && (b[1] != '$')
            && ((a[1] < '0') || (a[1] > '9'))
            && ((a[1] < 'A') || (a[1] > 'Z'))
            && ((a[1] < 'a') || (a[1] > 'z')))
          math_flag = 2;
        if ((PERL_flag != 0)
            && (b[1] != '$')
            && ((b[1] < 'A') || (b[1] > 'Z'))
            && ((b[1] < 'a') || (b[1] > 'z')))
          math_flag = 2;
      }
  }
/* If LaTeX is not used properly, then src2latex simply unset math_flag .          */
if ((math_flag != 0)
    && ((MAKE_flag != 0) || (PERL_flag != 0)
        || (SH_flag != 0) || (TCL_flag != 0)))
  {
    ptr = buf_ptr;
```

```
            tex_counter = 0;
            for (i = 0; i < (int)(FORMULA_LEN_MAX); ++i)
              {
                c0 = *ptr;
                ptr1 = inc_buf_ptr(ptr);
                c1 = *ptr1;
                ptr2 = inc_buf_ptr(ptr1);
                c2 = *ptr2;
                if ((c0 == '\n') && (c1 != '#'))
                  break;
                if ((c0 != '\\') && (c1 == '$') && (c2 <= ' '))
                  break;
                if ((c0 == '_') || (c0 == '^') || (c0 == '\\'))
                  ++tex_counter;
                ptr = ptr1;
              }
            if (i >= (int)(FORMULA_LEN_MAX))
              math_flag = 0;
            if ((c0 == '\n') && (c1 != '#'))
              math_flag = 0;
            if ((c0 != '\\') && (c1 == '$') && (c2 <= ' ')
                && (tex_counter == 0))
              math_flag = 0;
#ifdef DEBUGGING
            printf ("get_tex_flag(): math_flag is set to %d\n",
                    math_flag);
#endif
          }
        if ((b[1] <= ' ') && (b[0] == '\\') && (b[1] == '(')
            && (((b[2] >= '0') && (b[2] <= '9'))
                || ((b[2] >= 'a') && (b[2] <= 'z'))
                || ((b[2] == '\\') && (b[3] >= 'a') && (b[3] <= 'z'))
                || (b[2] == '{')))
          math_flag = 2;
        if ((b[1] <= ' ') && (b[0] == '\\') && (b[1] == '[')
            && (((b[2] >= '0') && (b[2] <= '9'))
                || ((b[2] >= 'a') && (b[2] <= 'z'))
                || ((b[2] == '\\') && (b[3] >= 'a') && (b[3] <= 'z'))
                || (b[2] == '{')))
          {
            math_flag = 2;
            ds_flag = 1;
          }
      }
#else
    if (math_flag == 0)
      {
        if ((a[1] != '\\') && (a[1] != '$') && (b[0] == '$'))
          {
            if (b[1] != '$')      /* check RCS keywords here                                  */
              {
                if ((b[1] == 'A') && (b[2] == 'u') && (b[3] == 't')
                    && (b[4] == 'h') && (b[5] == 'o') && (b[6] == 'r')
```

```
        && ((b[7] == ':') || (b[7] == '$')))
      rcs_flag = 2;
    if ((b[1] == 'D') && (b[2] == 'a') && (b[3] == 't')
        && (b[4] == 'e')
        && ((b[5] == ':') || (b[5] == '$')))
      rcs_flag = 2;
    if ((b[1] == 'H') && (b[2] == 'e') && (b[3] == 'a')
        && (b[4] == 'd') && (b[5] == 'e') && (b[6] == 'r')
        && ((b[7] == ':') || (b[7] == '$')))
      rcs_flag = 2;
    if ((b[1] == 'I') && (b[2] == 'd')
        && ((b[3] == ':') || (b[3] == '$')))
      rcs_flag = 2;
    if ((b[1] == 'L') && (b[2] == 'o') && (b[3] == 'c')
        && (b[4] == 'k') && (b[5] == 'e') && (b[6] == 'r')
        && ((b[7] == ':') || (b[7] == '$')))
      rcs_flag = 2;
    if ((b[1] == 'L') && (b[2] == 'o') && (b[3] == 'g')
        && ((b[4] == ':') || (b[4] == '$')))
      rcs_flag = 2;
    if ((b[1] == 'R') && (b[2] == 'e') && (b[3] == 'v')
        && (b[4] == 'i') && (b[5] == 's') && (b[6] == 'i')
        && (b[7] == 'o') && (b[8] == 'n')
        && ((b[9] == ':') || (b[9] == '$')))
      rcs_flag = 2;
    if ((b[1] == 'R') && (b[2] == 'C') && (b[3] == 'S')
        && (b[4] == 'f') && (b[5] == 'i') && (b[6] == 'l')
        && (b[7] == 'e')
        && ((b[8] == ':') || (b[8] == '$')))
      rcs_flag = 2;
    if ((b[1] == 'S') && (b[2] == 'o') && (b[3] == 'u')
        && (b[4] == 'r') && (b[5] == 'c') && (b[6] == 'e')
        && ((b[7] == ':') || (b[7] == '$')))
      rcs_flag = 2;
    if ((b[1] == 'S') && (b[2] == 't') && (b[3] == 'a')
        && (b[4] == 't') && (b[5] == 'e')
        && ((b[6] == ':') || (b[6] == '$')))
      rcs_flag = 2;
  }
if ( (BAS_flag == 0) && (PERL_flag == 0))
  {
    if (b[1] != '$')
      math_flag = 2;
  }
else                /* else-part takes care of BASIC or PERL string variables        */
  {
    if ((BAS_flag != 0)
        && (b[1] != '$')
        && ((a[1] < '0') || (a[1] > '9'))
        && ((a[1] < 'A') || (a[1] > 'Z'))
        && ((a[1] < 'a') || (a[1] > 'z')))
      math_flag = 2;
    if ((PERL_flag != 0)
```

```
                  && (b[1] != '$')
                  && ((b[1] < 'A') || (b[1] > 'Z'))
                  && ((b[1] < 'a') || (b[1] > 'z')))
                math_flag = 2;
            }
          /* If TeX is not used properly, then src2tex simply unset math_flag .        */
          if ((math_flag != 0)
              && ((MAKE_flag != 0) || (PERL_flag != 0)
                  || (SH_flag != 0) || (TCL_flag != 0)))
            {
              ptr = buf_ptr;
              tex_counter = 0;
              for (i = 0; i < (int)(FORMULA_LEN_MAX); ++i)
                {
                  c0 = *ptr;
                  ptr1 = inc_buf_ptr(ptr);
                  c1 = *ptr1;
                  ptr2 = inc_buf_ptr(ptr1);
                  c2 = *ptr2;
                  if ((c0 == '\n') && (c1 != '#'))
                    break;
                  if ((c0 != '\\') && (c1 == '$') && (c2 <= ' '))
                    break;
                  if ((c0 == '_') || (c0 == '^') || (c0 == '\\'))
                    ++tex_counter;
                  ptr = ptr1;
                }
              if (i >= (int)(FORMULA_LEN_MAX))
                math_flag = 0;
              if ((c0 == '\n') && (c1 != '#'))
                math_flag = 0;
              if ((c0 != '\\') && (c1 == '$') && (c2 <= ' ')
                  && (tex_counter == 0))
                math_flag = 0;
#ifdef DEBUGGING
              printf ("get_tex_flag(): math_flag is reset to %d\n",
                      math_flag);
#endif
            }
          if ((b[1] == '$') && (b[2] != '$'))
            {
              math_flag = 2;
              ds_flag = 1;
            }
        }
    }
#endif
    if ((math_flag == 0) && (esc_flag == 0))
      {
        if ((a[1] != '\\') && (b[0] == '{'))
          {
            if ((b[1] == '\\') || ((b[1] == '{') && (b[2] == '\\')))
              {
```

```
                    brace_counter = 1;
                    esc_flag = 2;
                    *buf_ptr = ' ';    /* replace a TEX-mode-start-brace with a space         */
                  }
                if ((b[1] == '\\') && (b[2] == ' '))
                  {
                    /* replace a TEX-mode-start-backslash with a blank space                  */
                    ptr = inc_buf_ptr(buf_ptr);
                    *ptr = '{';
                    ptr = inc_buf_ptr(ptr);
                    *ptr = '}';
                  }
                if ((b[1] == '{') && (b[2] == '\\') && (b[3] == ' '))
                  {
                    /* replace a TEX-mode-start-backslash with a blank space                  */
                    ptr = inc_buf_ptr(buf_ptr);
                    ptr = inc_buf_ptr(ptr);
                    *ptr = '{';
                    ptr = inc_buf_ptr(ptr);
                    *ptr = '}';
                  }
            }
      }
    if (rcs_flag != 0)
      tex_flag = (esc_flag != 0);
    else
      tex_flag = ((math_flag != 0) | (esc_flag != 0));
    return tex_flag;
}
```