

```

/*****
/*          E D I T . C          */
**-----**
/* Task      : Text Editor      */
**-----**
/* Authors   : Michael Tischer / Bruno Jennrich */
/* developed on : 05/03/1994      */
/* last update : 05/15/1994      */
/*****
#ifndef __EDIT_C          /* EDIT.C can also be #Included */
#define __EDIT_C

/*- Add include files -----*/
#include "edit.h"
#include "win.h"
#include "kbd.h"

/*- Remove the following rows within a project: -----*/
#include "win.c"

/*****
/* edit_GetRows : Get beginnings of rows to be displayed. */
**-----**
/* Input      : lpEdit      - Address of editor      */
/*              iStartRow    - first screen row to be recalculated */
/*              iEndRow      - last screen row to be recalculated */
/*              iStartBufPos - beginning of first screen row in text */
/* Output     : Number of recalculated rows          */
**-----**
/* Info : This function calculates the beginnings of the */
/*        rows only as long as necessary, i.e., if the */
/*        recalculated row is identical to the known row start */
/*        prior to calling this function, recalculation stops, */
/*        with the assumption that the rest of the text */
/*        is unchanged. */
/*****
int edit_GetRows( LPEDIT lpEdit,
                  int      iStartRow,
                  int      iEndRow,
                  int      iStartBufPos )
{ int i, j, iUsed, iChanged, iNumChanged;
  LPBYTE lpBuffer;

  iChanged = TRUE;

  lpEdit->iRowStarts[ iStartRow ] = iStartBufPos;

  for( i = iStartRow + 1; i < iEndRow; i++ )
    lpEdit->iRowStarts[ i ] = lpEdit->iUsed + 1;

  i = iStartRow;
  j = iStartBufPos;

  iUsed   = lpEdit->iUsed;
  lpBuffer = &lpEdit->lpBuffer[ j ];

  iNumChanged = 0;
  while( ( i <= iEndRow ) && ( j <= iUsed ) && iChanged )
  {
    j++;
    if( *(lpBuffer++) == '\n' )
    {
      i++;
      iChanged = ( lpEdit->iRowStarts[ i ] != j );
      if( iChanged ) iNumChanged++;
      lpEdit->iRowStarts[ i ] = j;
    }
  }
  return iNumChanged;
}

/*****
/* edit_LeftOffset : Determine first character of all rows to be */
/*                  displayed. */
**-----**
/* Input      : lpEdit - Address of Editor */
*/

```

```

/*          iRowPos - Address of an integer that receives the          */
/*          position of the insertion point within the                */
/*          current row.                                              */
/* Output   : TRUE - 'LeftOffset' has changed.                        */
/*          FALSE - 'LeftOffset' unchanged.                          */
/*-----**/
/* Info : In text mode the LeftOffset applies to all displayed      */
/*        rows of the Editor, since byte orientation exists.        */
/*        In graphical environments like Windows, the LeftOffset    */
/*        must be calculated for each row, using 'inch'              */
/*        or something similar as a measurement, not 'character'.   */
/*        (take proportional fonts into account!)                    */
/*****
int edit_LeftOffset( LPEDIT lpEdit, LPWORD iRowPos )
{ int iInc;
  /* Scroll screen contents by 1/3 width, but at least 10 characters */
  iInc = ((lpEdit->Win.iW)/3) > 10 ? 10 : (lpEdit->Win.iW)/3;
  if(!iInc) iInc = 1;

  /* Position of insertion point within the current row */
  *iRowPos = lpEdit->iEditPos - lpEdit->iRowStarts[ lpEdit->iEditRow ];

  /* Shift contents to the left or right? */
  if( ( *iRowPos - lpEdit->iLeftOffset ) >= lpEdit->Win.iW ) /* left */
  {
    lpEdit->iLeftOffset = ((*iRowPos-lpEdit->Win.iW)/iInc)*iInc+iInc;
    return TRUE;
  }

  if( *iRowPos < lpEdit->iLeftOffset ) /* right */
  {
    lpEdit->iLeftOffset = ((*iRowPos-iInc)/iInc)*iInc;
    if( lpEdit->iLeftOffset < 0 ) lpEdit->iLeftOffset = 0;
    return TRUE;
  }
  return FALSE;
}

/*****
/* edit_Display : Display current Editor contents.                  */
/*-----**/
/* Input   : lpEdit - Address of Editor                            */
/*          iStart - first screen row to display                    */
/*          iEnd   - last screen row to display                     */
/*****
void edit_Display( LPEDIT lpEdit, int iStart, int iEnd )
{ int i, n, iW, iRowPos, OldFlags;

  if( lpEdit->Win.uFlags & WIN_ACTIVE )
    lpEdit->Win.iAttr |= 0x08; /* show HighColor */
  else
    lpEdit->Win.iAttr &= ~0x08; /* hide Highcolor */

  OldFlags = lpEdit->Win.uFlags; /* save flags */
  lpEdit->Win.uFlags &= ~( WIN_CRLF | WIN_SCROLL | WIN_HASCURSORS );

  if( lpEdit->uFlags & EDIT_DISPLAY ) /* Editor only for display */
  {
    iStart = 0; /* display all rows */
    iEnd = lpEdit->Win.iH - 1;
    iRowPos = 0;
    OldFlags &= ~WIN_HASCURSORS;
  }
  else /* must screen contents be scrolled? */
  if( edit_LeftOffset( lpEdit, &iRowPos ) )
  {
    iStart = 0;
    iEnd = lpEdit->Win.iH - 1;
  }

  for( i = iStart; ( i <= iEnd ) && ( i < lpEdit->Win.iH ); i++ )
  {
    win_GotoXY( &lpEdit->Win, 0, i ); /* start screen row */

    n = lpEdit->iRowStarts[ i + 1 ]; /* last character of the row */

```

```

    if( n > lpEdit->iUsed ) n = lpEdit->iUsed;      /* last character? */
    /* iW = number of characters to be output in the row (i) */
    iW = ( n - lpEdit->iRowStarts[ i ] ) - lpEdit->iLeftOffset;

    if( iW <= 0 ) _win_Print( &lpEdit->Win, "\n", 1 );
    else
    {
        /* output maximum of one screen width */
        iW = iW > lpEdit->Win.iW ? lpEdit->Win.iW : iW;
        _win_Print( &lpEdit->Win,          /* output character */
                    &lpEdit->lpBuffer[ lpEdit->iRowStarts[ i ] +
                                        lpEdit->iLeftOffset ], iW );
        if( iW < lpEdit->Win.iW )          /* delete rest of row */
        {
            win_GotoXY( &lpEdit->Win, iW, i );
            _win_Print( &lpEdit->Win, "\n", 1 );
        }
    }
}

lpEdit->Win.uFlags = OldFlags;
win_GotoXY( &lpEdit->Win,          /* edit_LeftOffset() calculates iRowPos */
            iRowPos - lpEdit->iLeftOffset,
            lpEdit->iEditRow );
}

/*****
/* edit_SetBuffer : Initialize input buffer
**-----*/
/* Input      : lpEdit    - Address of Editor
/*              lpBuffer - Address of new Texted
/*              iBufSize - Number of characters in buffer
**-----*/
/* Info : After calling this function, the insertion point is
/*          at the beginning of the text specified by this function.
**-----*/
void edit_SetBuffer( LPEDIT lpEdit, LPBYTE lpBuffer, int iBufSize )
{ int i;

    /* new contents greater than Edit buffer ? */
    if( iBufSize > lpEdit->iSize ) iBufSize = lpEdit->iSize;

    for( i = 0; i < iBufSize; i++ )          /* Copy buffer contents */
        lpEdit->lpBuffer[ i ] = lpBuffer[ i ];

    lpEdit->iUsed = iBufSize;
    lpEdit->iEditPos = 0;
    lpEdit->iEditRow = 0;
    lpEdit->iLeftOffset = 0;

    edit_GetRows( lpEdit, 0, lpEdit->Win.iH, 0 );
    edit_Display( lpEdit, 0, lpEdit->Win.iH - 1 );
}

/*****
/* edit_init : Initialize Editor
**-----*/
/* Input      : iX, iY, iW, iH, iA - Window position
/*              -dimension
/*              -attribute
/*              lpEdit    - Address of Editor
/*              lpBuffer - Text memory to use
/*              iBufUsed - Characters already contained in text memory
/*              iBufSize - Size of text memory
/*              uFlags    - Editorflags : EDIT_DISPLAY - Display only
/*                          EDIT_NOCR    - No input
/*                          of 'CR'
**-----*/
/* Info : The text memory for receiving the input characters must be
/*          made available "from outside" .
**-----*/
void edit_Init( int iX, int iY, int iW, int iH, int iA,
                LPEDIT lpEdit, LPBYTE lpBuffer,
                int iBufUsed,
                int iBufSize,
                int uFlags )
{ int i;
  UINT winFlags = WIN_SCROLL | WIN_CRLF;

```

```

if( !( lpEdit->uFlags & EDIT_DISPLAY ) )
    winFlags |= WIN_HASCURSOR;

win_Init( &lpEdit->Win, iX, iY, iW, iH, iA, winFlags );

lpEdit->lpBuffer      = lpBuffer;
lpEdit->iUsed          = iBufUsed;
lpEdit->iSize          = iBufSize;
lpEdit->iEditPos       = 0;
lpEdit->iEditRow       = 0;
lpEdit->iLeftOffset    = 0;
lpEdit->uFlags        = uFlags;

edit_GetRows( lpEdit, 0, iH, 0 );          /* Format text... */
edit_Display( lpEdit, 0, iH - 1 );        /* and display */
}

/*****
/* edit_PreviousRow : Get previous row of the first screen
/*
/* row.
**-----**
/* Input   : lpEdit - Address of Editor
/* Output  : TRUE - Previous row exists
/*          FALSE - First screen row = first Editor row
**-----**
/* Info : As long as a row is bounded by an '\n', the location
/* of the previous row is to be determined by searching
/* in the text. If WordWrap features are built in, an array
/* must include the beginning of all previous rows, from which
/* the beginning of the previous row can easily be read
/* out.
**-----**
int edit_PreviousRow( LPEDIT lpEdit )
{ int iPrevRow, i;

    /* Start of the first row in text */
    iPrevRow = lpEdit->iRowStarts[ 0 ];

    if( iPrevRow ) /* Does first screen row begin at start of text? */
    {
        iPrevRow--; /* no, previous row must exist */

        for( i = lpEdit->Win.iH + 1; i > 0; i-- ) /* Scroll screen row
            lpEdit->iRowStarts[ i ] = /* starts (internal)
            lpEdit->iRowStarts[ i - 1 ]; /* down.

        while( ( iPrevRow > 0 ) && /* Find character before '\n' ... */
            ( lpEdit->lpBuffer[ iPrevRow - 1 ] != '\n' ) )
            iPrevRow--;

        /* and set as new start of first screen row */
        lpEdit->iRowStarts[ 0 ] = iPrevRow;
        return TRUE;
    }
    return FALSE;
}

/*****
/* edit_NextRow : Find next row of the last screen row
**-----**
/* Input   : lpEdit - Address of Editor
/* Output  : TRUE - Next row exists
/*          FALSE - last Editor row already being displayed
/*          (doesn't have to be last screen row!)
**-----**
/* Info : Screen rows that aren't used (e.g., with short texts)
/* "begin" at last character of the text (see iRowStarts)
**-----**
int edit_NextRow( LPEDIT lpEdit )
{ int i;

    /* last screen row != Text end ? */
    if( lpEdit->iRowStarts[ lpEdit->Win.iH ] <= lpEdit->iUsed )
    {
        /* Scroll rows (internally) up */
        for( i = 0 ; i <= lpEdit->Win.iH + 1; i++ )
            lpEdit->iRowStarts[ i ] = lpEdit->iRowStarts[ i + 1 ];
        /* calculate next row start */
    }
}

```

```

        edit_GetRows( lpEdit, lpEdit->Win.iH - 1, lpEdit->Win.iH + 1,
                      lpEdit->iRowStarts[ lpEdit->Win.iH - 1 ] );
    }
    return TRUE;
}
return FALSE;
}

/*****
/* edit_StartOfText : Display first row of text and place insertion
/*
/* point at start of text
**-----*/
/* Input      : lpEdit - Address of Editor
/* Output     : TRUE - Screen must be redisplayed
/*             FALSE - First text row already being displayed
*****/
int edit_StartOfText( LPEDIT lpEdit )
{
    if( lpEdit->iEditPos != 0 )
    {
        lpEdit->iEditPos = 0;
        lpEdit->iEditRow = 0;
        if( lpEdit->iRowStarts[ 0 ] != 0 )
            edit_GetRows( lpEdit, 0, lpEdit->Win.iH, 0 );
        edit_Display( lpEdit, 0, lpEdit->Win.iH - 1 );
        return TRUE;
    }
    return FALSE;
}

/*****
/* edit_EndOfText : Display last row of text and move insertion point
/*
/* to last character.
**-----*/
/* Input      : lpEdit - Address of Editor
/*             iDisplay - Display changes also
/* Output     : TRUE - Screen needs to be redisplayed
/*             FALSE - Last text row already being displayed
*****/
int edit_EndOfText( LPEDIT lpEdit, int iDisplay )
{
    int j;
    /* Is insertion point already on the last character? */
    if( lpEdit->iEditPos != lpEdit->iUsed )
    {
        /* Last text row already being displayed? */
        for( j = 0; j < lpEdit->Win.iH; j++ )
            if( lpEdit->iRowStarts[ j ] > lpEdit->iUsed )
            {
                /* Find last text row on the screen and move insertion
                /* point and current row to this screen row
                lpEdit->iEditRow = j - 1;
                lpEdit->iEditPos = lpEdit->iUsed;
                if( lpEdit->iEditRow < 0 ) lpEdit->iEditRow = 0;
                if( iDisplay ) edit_Display( lpEdit, 0, lpEdit->Win.iH );
                return TRUE;
            }

            /* last character = start of all screen rows */
        for( j = 0; j <= lpEdit->Win.iH + 1; j++ )
            lpEdit->iRowStarts[ j ] = lpEdit->iUsed + 1;

        /* lpEdit->Win.iH-1 scroll rows up */
        j = 0;
        lpEdit->iRowStarts[ 0 ] = lpEdit->iUsed;
        while( edit_PreviousRow( lpEdit ) & ( j < lpEdit->Win.iH - 1 ) )
            j++;

        lpEdit->iEditRow = j;
        lpEdit->iEditPos = lpEdit->iUsed;
        if( iDisplay ) edit_Display( lpEdit, 0, lpEdit->Win.iH );
        return TRUE;
    }
    return FALSE;
}

/*****
/* edit_SetRowPos : Place insertion point in current edit row
/*
/* (iEditRow) at specified position
**-----*/
/* Input      : lpEdit - Address of Editor
**-----*/

```

```

/*          iRowPos - Column where insertion point is to be          */
/*          placed in current edit row.                              */
/*****
void edit_SetRowPos( LPEDIT lpEdit, int iRowPos )
{
    /* Does new row even have iRowPos characters? */
    if( ( lpEdit->iRowStarts[ lpEdit->iEditRow + 1 ] -
        lpEdit->iRowStarts[ lpEdit->iEditRow ] ) > iRowPos )
        lpEdit->iEditPos =
            lpEdit->iRowStarts[ lpEdit->iEditRow ] + iRowPos;
    else /* No, then place on last character of the row */
        lpEdit->iEditPos =
            lpEdit->iRowStarts[ lpEdit->iEditRow + 1 ] - 1;
}

/*****
/* edit_CRSRUp : Move insertion point one row up          */
/*-----*/
/* Input      : lpEdit      - Address of Editor          */
/*             iUpdatePos - Place insertion point in new row on old      */
/*             column                                             */
/* Output     : TRUE      - Insertion point was moved      */
/*             FALSE     - Insertion point is in first text row */
/*****
int edit_CRSRUp( LPEDIT lpEdit, int iUpdatePos )
{ int iRowPos, iStart, iEnd;

    iStart = 0; /* Rows to be redisplayed */
    iEnd = 0;
    /* Get position of insertion point within the current row */
    iRowPos = lpEdit->iEditPos - lpEdit->iRowStarts[ lpEdit->iEditRow ];
    /* Scroll screen contents during text display! */
    if( lpEdit->uFlags & EDIT_DISPLAY ) lpEdit->iEditRow = 0;
    /* Current row != Text start ? */
    if( lpEdit->iRowStarts[ lpEdit->iEditRow ] > 0 )
    {
        lpEdit->iEditRow--; /* Decrement current row */
        if( lpEdit->iEditRow < 0 ) /* Scroll text down */
        {
            edit_PreviousRow( lpEdit );
            lpEdit->iEditRow = 0;
            iEnd = lpEdit->Win.iH - 1; /* Redisplay all screen rows */
        }
        /* Place insertion point in new row at same position */
        if( iUpdatePos ) edit_SetRowPos( lpEdit, iRowPos );
        edit_Display( lpEdit, iStart, iEnd );
        return TRUE;
    }
    return FALSE;
}

/*****
/* edit_CRSRDn : Move insertion point one row down          */
/*-----*/
/* Input      : lpEdit      - Address of Editor          */
/*             iUpdatePos - Place insertion point in new row at old      */
/*             column                                             */
/* Output     : TRUE      - Insertion point was moved      */
/*             FALSE     - Insertion point is in last text row */
/*****
int edit_CRSRDn( LPEDIT lpEdit, int iUpdatePos )
{ int iRowPos, iStart, iEnd;

    iStart = 0; /* Rows to be redisplayed */
    iEnd = 0;
    /* Get position of insertion point within the current row */
    iRowPos = lpEdit->iEditPos - lpEdit->iRowStarts[ lpEdit->iEditRow ];
    /* Scroll screen contents during text display! */
    if( lpEdit->uFlags & EDIT_DISPLAY )
        while((lpEdit->iRowStarts[lpEdit->iEditRow+1]<lpEdit->iUsed) &&
            (lpEdit->iEditRow<(lpEdit->Win.iH-1))) lpEdit->iEditRow++;

    if( lpEdit->iRowStarts[ lpEdit->iEditRow + 1 ] < lpEdit->iUsed )
    {
        lpEdit->iEditRow++;
        if( lpEdit->iEditRow == lpEdit->Win.iH )
        {

```

```

        iStart = 0;
        iEnd = lpEdit->Win.iH - 1;
        edit_NextRow( lpEdit );
        lpEdit->iEditRow = lpEdit->Win.iH - 1;
    }
    if( iUpdatePos ) edit_SetRowPos( lpEdit, iRowPos );
    edit_Display( lpEdit, iStart, iEnd );
    return TRUE;
}
return FALSE;
}

/*****
/* edit_CRSRLeft : Move insertion point one character to the left */
/*-----*/
/* Input      : lpEdit      - Address of Editor */
/* Output     : TRUE - Insertion point was moved */
/*             FALSE - Insertion point is at start of text */
*****/
int edit_CRSRLeft( LPEDIT lpEdit )
{
    /* On Display-Only move screen contents */
    if( lpEdit->uFlags & EDIT_DISPLAY )
    {
        if( lpEdit->iLeftOffset > 0 ) lpEdit->iLeftOffset--;
        edit_Display( lpEdit, 0, 0 );
        return TRUE;
    }

    if( lpEdit->iEditPos > 0 )
    {
        lpEdit->iEditPos--;
        if( lpEdit->iEditPos < lpEdit->iRowStarts[ lpEdit->iEditRow ] )
            edit_CRSRUp( lpEdit, FALSE );
        edit_Display( lpEdit, 0, 0 );
        return TRUE;
    }
    return FALSE;
}

/*****
/* edit_CRSRRight : Move insertion point one character to the right */
/*-----*/
/* Input      : lpEdit      - Address of Editor */
/* Output     : TRUE - Insertion point was moved */
/*             FALSE - Insertion point is at end of text */
*****/
int edit_CRSRRight( LPEDIT lpEdit )
{
    /* On Display-Only move screen contents */
    if( lpEdit->uFlags & EDIT_DISPLAY )
    {
        lpEdit->iLeftOffset++;
        edit_Display( lpEdit, 0, 0 );
        return TRUE;
    }

    if( lpEdit->iEditPos < lpEdit->iUsed )
    {
        lpEdit->iEditPos++;
        if( lpEdit->iEditPos >= lpEdit->iRowStarts[ lpEdit->iEditRow + 1 ] )
            edit_CRSRDn( lpEdit, FALSE );
        edit_Display( lpEdit, 0, 0 );
        return TRUE;
    }
    return FALSE;
}

/*****
/* edit_Home : Place insertion point at beginning of row */
/*-----*/
/* Input      : lpEdit      - Address of Editor */
/*             iDisplay     - != 0 Display change immediately */
/*             == 0 Do not display changes */
*****/
void edit_Home( LPEDIT lpEdit, int iDisplay )

```

```

{
    /* On Display-Only LeftOffset to row start */
    if( lpEdit->uFlags & EDIT_DISPLAY )
    {
        lpEdit->iLeftOffset = 0;
        if( iDisplay ) edit_Display( lpEdit, 0, 0 );
    }

    if( lpEdit->iEditPos > 0 ) /* Place insertion point at row start */
    {
        lpEdit->iEditPos = lpEdit->iRowStarts[ lpEdit->iEditRow ];
        if( iDisplay ) edit_Display( lpEdit, 0, 0 );
    }
}

/*****
/* edit_End : Place insertion point at end of row
**-----**
/* Input : lpEdit - Address of Editor
**-----**
void edit_End( LPEDIT lpEdit )
{
    if( lpEdit->iEditPos < lpEdit->iUsed )
    {
        lpEdit->iEditPos = lpEdit->iRowStarts[ lpEdit->iEditRow + 1 ] - 1;
        edit_Display( lpEdit, 0, 0 );
    }
}

/*****
/* edit_CRSRPgDn : Move insertion point one screen row
/* down
**-----**
/* Input : lpEdit - Address of Editor
**-----**
int edit_CRSRPgDn( LPEDIT lpEdit )
{ int i, j, iH, iRowPos;

    /* Get position of insertion point within the current row */
    iRowPos = lpEdit->iEditPos - lpEdit->iRowStarts[ lpEdit->iEditRow ];
    iH = ( lpEdit->Win.iH - 1 ); /* Number of rows to scroll */
    if( !iH ) iH = 1; /* Note single rows! */

    for( i = 0; i < iH; i++ ) /* Get new rows */
        if( edit_NextRow( lpEdit ) ) j++;

    if( ( !j ) && /* End of text already displayed ? */
        ( lpEdit->iRowStarts[ lpEdit->iEditRow + 1 ] > lpEdit->iUsed ) )
        return FALSE;

    if( j < iH ) /* Move edit row as far as possible */
        while( ( j < lpEdit->Win.iH ) &&
            ( lpEdit->iRowStarts[ j + 1 ] < lpEdit->iUsed ) )
            j++;
    else j = lpEdit->iEditRow; /* Contents were scrolled by one page,
/* so the edit row remains on the same
/* screen row.

    lpEdit->iEditRow = j;
    edit_SetRowPos( lpEdit, iRowPos );
    edit_Display( lpEdit, 0, lpEdit->Win.iH - 1 );
    return TRUE;
}

/*****
/* edit_CRSRPgUp : Move insertion point one screen row
/* up
**-----**
/* Input : lpEdit - Address of Editor
**-----**
int edit_CRSRPgUp( LPEDIT lpEdit )
{ int i, j, iH, iRowPos;

    /* Get position of insertion point within the current row */
    iRowPos = lpEdit->iEditPos - lpEdit->iRowStarts[ lpEdit->iEditRow ];
    iH = ( lpEdit->Win.iH - 1 ); /* Number of rows to scroll */
    if( !iH ) iH = 1; /* Note single rows! */

```

```

for( i = 0; i < iH; i++ ) /* Get new rows */
    if( edit_PreviousRow( lpEdit ) ) j++;

if( !( j + lpEdit->iEditRow ) ) return FALSE; /* No movement? */

if( j < iH ) j = 0; /* New current edit row */
else j = lpEdit->iEditRow;

lpEdit->iEditRow = j;
edit_SetRowPos( lpEdit, iRowPos );
edit_Display( lpEdit, 0, lpEdit->Win.iH );
return TRUE;
}

/*****
/* edit_AdjustRow : Find the right screen row for edit position */
-----*/
/* Input : lpEdit - Address of Editor */
/* Output : TRUE - Edit row being displayed on screen */
/* FALSE - Edit row is not displayed */
*****/
int edit_AdjustRow( LPEDIT lpEdit )
{
    int i;
    for( i = 0; i < lpEdit->Win.iH - 1; i++ ) /* All screen rows */
    {
        if( ( lpEdit->iEditPos >= lpEdit->iRowStarts[ i ] ) &&
            ( lpEdit->iEditPos < lpEdit->iRowStarts[ i + 1 ] ) )
        {
            lpEdit->iEditRow = i;
            return TRUE;
        }
    }
    if( lpEdit->iEditPos >= lpEdit->iRowStarts[ lpEdit->Win.iH - 1 ] )
        lpEdit->iEditRow = lpEdit->Win.iH;
    else
        if( lpEdit->iEditPos < lpEdit->iRowStarts[ 0 ] )
            lpEdit->iEditRow = -1;
    return FALSE;
}

/*****
/* edit_DeleteChar : Delete character under insertion point */
-----*/
/* Input : lpEdit - Address of Editor */
/* Output : TRUE - Character was deleted */
/* FALSE - Insertion point at end of text, delete not possible */
*****/
int edit_DeleteChar( LPEDIT lpEdit )
{
    int i, iStart, iEnd; /* On Display-Only no deletion */
    if( lpEdit->uFlags & EDIT_DISPLAY ) return TRUE;

    if( lpEdit->iEditPos < lpEdit->iUsed )
    {
        /* Remove character from buffer, or "press together" buffer */
        for( i = lpEdit->iEditPos; i < lpEdit->iUsed; i++ )
            lpEdit->lpBuffer[ i ] = lpEdit->lpBuffer[ i + 1 ];
        lpEdit->iUsed--; /* Decrement number of characters in buffer */
        /* Display rows after current row again */
        for( i = lpEdit->iEditRow + 1; i < lpEdit->Win.iH + 1; i++ )
            lpEdit->iRowStarts[ i ] -= 1;

        iStart = lpEdit->iEditRow;
        iEnd = lpEdit->iEditRow;
        /* Calculate how many rows have to be redisplayed */
        iEnd += edit_GetRows( lpEdit,
                             lpEdit->iEditRow,
                             lpEdit->Win.iH + 1,
                             lpEdit->iRowStarts[ lpEdit->iEditRow ] ) + 1;
        edit_Display( lpEdit, iStart, iEnd );
        return TRUE;
    }
    return FALSE;
}

```

```

/*****
/* edit_DeleteArea : Delete text area */
**-----**
/* Input      : lpEdit - Address of Editor */
/*              iStart - first character to delete */
/*              iLen   - number of characters to delete */
/* Output     : TRUE   - area was deleted */
/*              FALSE  - area not completely in text, delete not */
/*                  possible */
*****/
int edit_DeleteArea( LPEDIT lpEdit, int iStart, int iLen )
{
    int i, iEnd;

    /* area to be deleted completely in text? */
    if( iStart + iLen <= lpEdit->iUsed )
    {
        iEnd = lpEdit->iUsed - iLen; /* new end of text */
        for( i = iStart; i < iEnd; i++ ) /* "press together" buffer */
            lpEdit->lpBuffer[ i ] = lpEdit->lpBuffer[ i + iLen ];
        lpEdit->iUsed = iEnd; /* set new buffer size */

        /* insertion point after area to be deleted ? */
        if( lpEdit->iEditPos >= iStart )
            if( lpEdit->iEditPos <= iStart + iLen )
                lpEdit->iEditPos = iStart; /* insertion point in area */
            else
                lpEdit->iEditPos -= iLen; /* insertion point after */

        if( iStart < lpEdit->iRowStarts[ 0 ] ) /* update beginning of */
            lpEdit->iRowStarts[0] -= iLen; /* first screen row */

        iStart = 0; /* Calculate number of rows to redisplay */
        iEnd += edit_GetRows( lpEdit,
                               0,
                               lpEdit->Win.iH + 1,
                               lpEdit->iRowStarts[ 0 ] );
        edit_AdjustRow( lpEdit ); /* get current insertion point */
        edit_Display( lpEdit, iStart, iEnd );
        return TRUE;
    }
    return FALSE;
}

/*****
/* edit_DeleteFirstRow : Delete first row of text */
**-----**
/* Input      : lpEdit - Address of Editor */
/*              iDisplay - Display changes */
/* Output     : TRUE   - Able to delete first row */
/*              FALSE  - No more characters there to delete! */
*****/
int edit_DelFirstRow( LPEDIT lpEdit, int iDisplay )
{
    int i;
    i = 0;
    while( ( lpEdit->lpBuffer[ i ] != '\n' ) && ( i < lpEdit->iUsed ) )
        i++;
    if( lpEdit->lpBuffer[ i ] == '\n' ) i++;
    if( !edit_DeleteArea( lpEdit, 0, i ) ) return FALSE;
    if( iDisplay ) edit_Display( lpEdit, 0, lpEdit->Win.iH );
    return TRUE;
}

/*****
/* edit_Insert : Insert text at insertion point */
**-----**
/* Input      : lpEdit - Address of Editor */
/*              lpBuffer - Address of text to be inserted */
/*              iBufSize - Number of characters to be inserted */
/* Output     : TRUE   - Able to insert characters */
/*              FALSE  - Text buffer too small */
*****/
int edit_Insert( LPEDIT lpEdit, LPBYTE lpBuffer, int iBufSize )
{
    int i, iEnd, iStart;
    if( lpEdit->iUsed + iBufSize < lpEdit->iSize )
    {
        for( i = lpEdit->iUsed + iBufSize; i > lpEdit->iEditPos; i-- )
            lpEdit->lpBuffer[ i ] = lpEdit->lpBuffer[ i - iBufSize ];
    }
}

```

```

/* "fit in" new text */
for( i = 0; i < iBufSize; i++ )
    lpEdit->lpBuffer[ lpEdit->iEditPos + i ] = lpBuffer[ i ];

lpEdit->iUsed += iBufSize; /* Adjust Editpos and size of */
lpEdit->iEditPos += iBufSize; /* text buffer */
iStart = lpEdit->iEditRow;
iEnd = lpEdit->iEditRow + 1;

/* Recalculate starts of screen rows */
for( i = lpEdit->iEditRow + 1; i <= lpEdit->Win.iH; i ++ )
    if( lpEdit->iRowStarts[ i ] <= lpEdit->iUsed )
        lpEdit->iRowStarts[ i ] += iBufSize;

iEnd = edit_GetRows( lpEdit,
                    iStart,
                    lpEdit->Win.iH + 1,
                    lpEdit->iRowStarts[ lpEdit->iEditRow ] ) + 1;

iEnd += iStart;

edit_AdjustRow( lpEdit ); /* get current insertion point */
/* new row through insertion? */
if( lpEdit->iEditRow == lpEdit->Win.iH )
{
    edit_NextRow( lpEdit );
    iStart = 0;
    iEnd = lpEdit->Win.iH - 1;
    lpEdit->iEditRow = lpEdit->Win.iH - 1;
}
edit_Display( lpEdit, iStart, iEnd );
return TRUE;
}
return FALSE;
}

/*****
/* _edit_Append : Add text to end of existing text. */
/* Do not display changes */
*****/
/* Input : lpEdit - Address of Editor */
/* lpBuffer - Address of text to be appended */
/* iBufSize - Number of characters to be appended */
/* Output : TRUE - Able to insert characters */
/* FALSE - Text buffer too small */
*****/
int _edit_Append( LPEDIT lpEdit, LPBYTE lpBuffer, int iBufSize )
{
    if( lpEdit->iUsed + iBufSize < lpEdit->iSize )
    {
        int i;

        /* Append text */
        for( i = 0; i < iBufSize; i++ )
            lpEdit->lpBuffer[ lpEdit->iUsed++ ] = lpBuffer[ i ];
        edit_GetRows( lpEdit, 0,
                    lpEdit->Win.iH + 2,
                    lpEdit->iRowStarts[ 0 ] );
        edit_AdjustRow( lpEdit );
        return TRUE;
    }
    return FALSE;
}

/*****
/* edit_Append : Append text to end of existing text. */
/* If buffer is too small, delete first row! */
*****/
/* Input : lpEdit - Address of Editor */
/* lpBuffer - Address of text to be appended */
/* iBufSize - Number of characters to append */
/* < 0 : searching for end of string */
/* Output : TRUE - Able to insert characters */
/* FALSE - Text buffer too small */
*****/
int edit_Append( LPEDIT lpEdit, LPBYTE lpBuffer, int iBufSize )
{
    if( iBufSize < 0 )
    {

```

```

    iBufSize = 0;
    while( lpBuffer[ iBufSize ] ) iBufSize++;
}

while( !_edit_Append( lpEdit, lpBuffer, iBufSize ) )
    if( !_edit_DelFirstRow( lpEdit, FALSE ) ) return FALSE;
edit_EndOfText( lpEdit, FALSE );
edit_Home( lpEdit, TRUE );
return TRUE;
}

/*****
/* edit_Leave : Editor loses edit focus */
**-----*/
/* Input      : lpEdit - Address of Editor */
/*****
void edit_Leave( LPEDIT lpEdit )
{
    lpEdit->Win.uFlags &= ~WIN_ACTIVE;          /* Clear ACTIVE flag */
    edit_Display( lpEdit, 0, lpEdit->Win.iH );    /* display all rows */
}

/*****
/* edit_Enter : Editor receives edit focus */
**-----*/
/* Input      : lpEdit - Address of Editor */
/*****
void edit_Enter( LPEDIT lpEdit )
{
    lpEdit->Win.uFlags |= WIN_ACTIVE;          /* set ACTIVE flag */
    edit_Display( lpEdit, 0, lpEdit->Win.iH );    /* display all rows */
}

/*****
/* edit_ProcessChar : Process key strokes */
**-----*/
/* Input      : lpEdit - Address of Editor */
/*              uChar - Key code (also enhanced > 256) */
/*****
void edit_ProcessChar( LPEDIT lpEdit, UINT uChar )
{ int Beep;
  BYTE bIns;

  bIns = LOBYTE( uChar );                      /* character to be inserted */
  Beep = FALSE;

  switch( uChar )
  {
      case KBD_UP:      Beep = !edit_CRSRUp( lpEdit, TRUE );    break;
      case KBD_LEFT:    Beep = !edit_CRSRLeft( lpEdit );        break;
      case KBD_DN:      Beep = !edit_CRSRDn( lpEdit, TRUE );    break;
      case KBD_RIGHT:   Beep = !edit_CRSRRight( lpEdit );        break;
      case KBD_DEL:     Beep = !edit_DeleteChar( lpEdit );      break;
      case KBD_HOME:    edit_Home( lpEdit, TRUE );              break;
      case KBD_END:     edit_End( lpEdit );                      break;
      case KBD_PGUP:    Beep = !edit_CRSRPgUp( lpEdit );        break;
      case KBD_PGDN:    Beep = !edit_CRSRPgDn( lpEdit );        break;
      case KBD_CTRLPGDN: Beep = !edit_EndOfText( lpEdit, TRUE ); break;
      case KBD_CTRLPGUP: Beep = !edit_StartOfText( lpEdit );    break;
      case KBD_BACK:
          if( ! ( lpEdit->uFlags & EDIT_DISPLAY ) )
          {
              Beep = !edit_CRSRLeft( lpEdit );
              if( !Beep ) Beep = !edit_DeleteChar( lpEdit );
              if( Beep ) win_Beep();
          }
          break;
      case KBD_LF:
      case KBD_CR:
          if( lpEdit->uFlags & EDIT_NOCR ) break;
          bIns = '\n';
      default:
          if( ( ! ( lpEdit->uFlags & EDIT_DISPLAY ) ) && bIns )
              edit_Insert( lpEdit, &bIns, 1);
  }
  if( Beep ) win_Beep();
}

```

```

}

/*****
/* control_leave : Control and associate object lose          */
/*                  edit focus                                */
/*-----*/
/* Input      : lpWin - Address of window in which control is */
/*                  displayed                                  */
/*                  lpControl - Address of control             */
*****/
void control_Leave( LPWINDOW lpWin, LPCONTROL lpControl )
{
    UINT OldFlags;
    switch( lpControl->bType )
    {
        case CTL_EDIT:
            edit_Leave( ( LPEDIT )lpControl->lpAdditional );
            break;
    }

    OldFlags = lpWin->uFlags;
    lpWin->uFlags &= ~WIN_ACTIVE;
    lpWin->iAttr &= ~0x08;
    win_PrintAt( lpWin, lpControl->iX, lpControl->iY, lpControl->lpText );
    lpWin->uFlags = OldFlags;
}

/*****
/* control_enter : Control and associate object receive edit focus */
/*-----*/
/* Input      : lpWin - Address of window in which control is */
/*                  displayed                                  */
/*                  lpControl - Address of control             */
*****/
void control_Enter( LPWINDOW lpWin, LPCONTROL lpControl )
{
    UINT OldFlags;

    OldFlags = lpWin->uFlags;
    lpWin->uFlags |= WIN_ACTIVE;
    lpWin->iAttr |= 0x08;
    win_PrintAt( lpWin, lpControl->iX, lpControl->iY, lpControl->lpText );
    lpWin->uFlags = OldFlags ;

    switch( lpControl->bType )
    {
        case CTL_EDIT:
            edit_Enter( ( LPEDIT )lpControl->lpAdditional );
            break;
    }
}

/*****
/* control_ProcessChar : Control and associate object receive */
/*                  key code                                  */
/*-----*/
/* Input      : lpControl - Address of control                */
/*                  uChar      - key code to be processed      */
*****/
void control_ProcessChar( LPCONTROL lpControl, UINT uChar )
{
    switch( lpControl->bType )
    {
        case CTL_EDIT:
            edit_ProcessChar( lpControl->lpAdditional, uChar );
    }
}
#endif

```