SHPV Library Function Manual for Application Program

Display functions ....

LibLine, LibMeshLine, LibLineClr, LibLineCplmnt, LibDotOn, LibDotOff, LibPutProFont, LibGetProFontSize, LibPutProStr, LibGetProStrSize, LibPut35Font, LibPut35Str, LibReverse, LibMesh, LibBox, LibPutGraph, LibPutGraphM, LibPutGraphO, LibPutFarData, LibGetGraph, LibGrpUp, LibGrpDwn, LibPutDisp, LibPutDispBox, LibClrDisp, LibClrBox, LibSetDispMode, LibInitDisp, LibSKeyRevSub, LibGdsBox, LibGdsBoxMesh, LibGdsBoxClr, LibGdsBoxCmp, LibGdsClr, LibGdsReverse, LibGdsMesh, LibGdsDotOn, LibGdsDotOff, LibGdsDotCmp, LibGdsLine, LibGdsLineClr, LibGdsLineMesh, LibGdsLineCmp, LibPutBoxSub, LibCngeBoxSub, LibPutDotSub, LibPutLineSub, LibGrphUpSideDown, LibStringDsp

Window functions ....

LibOpenWindow, LibOpenWindowS, LibCloseWindow,

Touch functions ....

LibTchInit, LibTchStackClr, LibTchStackPop, LibTchStackPush, LibTchHardIcon, LibTchWait, LibIconPrint, LibIconPrintR, LibIconPrintM, LibIconClick, LibIconClick2, LibScrollPrint, LibScrollArrowPrint, LibScrollClick, LibScrPosCheck, LibKeyInit, LibDispKey, LibGetKeyM, LibCldKeyInit, LibGetCale, LibInputTime, LibInputTimeBar, LibInputTerm, LibSKeyRev, LibSKeyIsCd, LibIconMoveDown, LibIconMoveUp, LibBkSampleInit, LibBkSampleCheck, LibBkSampleInitSub, LibBlockIconClick, LibRepOff, *LibTchWaitScan*

FLASH functions ....

LibFileFindNext, LibFileFindPrev, LibFileFindNextExt, LibNextSearchCld, LibFileRead, LibFileWrite, LibFileCorect, LibFileRemove, LibFileRemoveAll, LibGetFileInfo, LibGetFileCnt, LibGetFlash, LibGetFreeBlock, LibGetDataCond, LibFileRemake, LibTodoFileRemove, LibFileExch, LibTelPtCnvrt, LibFileWriteCheckInit, LibFileWriteCheck, LibFileReadEx

Menu functions

LibWinIcnMsg, LibSelWindow, LibSelWindow2, LibSelWindowExt, LibSelWinExt2A, LibSelWinExt2B, LibPullDown, LibPullDownInit, LibPullDownAtrSet, LibEditPullDown, LibSelWinLckA, LibSelWinLckB, LibSelectFont, *LibSelWinOpen2, LibSelectWin, LibSelWinTchSet*

System functions

LibSaveSysRam, LibSaveSysRamB, LibGetBLD, LibGetVersion, LibELHandle, LibGetEL, LibGetLang, LibSetLang, LibSoundGet, LibSoundSet, LibContrastInit, LibContrastUp, LibContrastDown, LibDigitizer, LibPassClr, LibPassSet, LibPassGet, LibPassChk, LibGetAPOTime, LibSetAPOTime, LibSetKeyKind, LibGetKeyKind, LibBuzzerOff, LibBuzzerOn, LibGetLangInf, *SysGetPONstat, SysSetPONstat, SysGetSUPstat, SysSetSUPstat, SysGetELTime, SysSetELTime, LibGetCommDevice*

Function functions

LibFuncDateTime, LibFuncSound, LibFuncFormat, LibFuncLang, LibFuncCapa, LibFuncContrast, LibFuncDigitizer, LibFuncMemoryManagement, LibFuncPtool, LibCalWin, *LibFuncUSB*

Calculator functions

LibCalBase, LibCalBaseData, LibCalRoot, LibCalKeyInit, LibCalKeyDsp, LibCalKeyTchWait, LibCalBuf2Dat, LibCalDat2Buf, *LibCalBase2*

Debug functions

LibPutMsgDlg, LibPutMsgDlg2, LibPutMsgDlg3, LibPutMsgDlg4

ADDIN functions

LibExeAddin, LibGetDLAllNum, LibGetUserMode, LibGetProgramName LibGetModeVer, LibGetLibVer, LibGetMenuIcon, LibGetListIcon LibCheckPMode, LibSubEntrySave, LibSubEntryDel, LibSubEntryRename LibSubEntrySearch, LibGetSubEntrySt, LibGetSubEntNum, LibGetAllEntry

FLASH functions (Call far pointer of the file buffer)

LibLFileFindNext, LibLFileFindPrev, LibLFileFindNextExt, LibLNextSearchCld, LibLFileRead, LibLFileWrite, LibLFileCorect, LibLFileRemove, LibLFileRemoveAll, LibLGetFileInfo, LibLGetFileCnt, LibLTodoFileRemove, LibLFileExch, LibLFileWriteCheck, LibLFileReadEx

- Display functions -

[Function name] `LibLine`

[Syntax]
```
#include "define.h"
#include "libc.h"
byte LibLine(int x, int y, int xsize, int ysize, byte bold)
```

[Arguments]
```
int      x        :IN      Start coordinate - Horizontal
int      y        :IN      Start coordinate - Vertical
int      xsize    :IN      Border size - width     (Min==1)
int      ysize    :IN      Border size - height    (Min==1)
byte     bold     :IN      Line weight      (Min==1)
```

[Return values]
```
byte     status   0:  Argument error
                  1:  Normal
```

[Description] Draws a border.
This is used to draw a border with a size specified by "xsize" and "ysize" from the start coordinate specified by (x, y).

[Examples of usage]
```
LibLine(10, 227, 304, 1,1);
LibPutDisp();
```

- Display functions -

[Function name] `LibMeshLine`

[Syntax]

```
    #include  "define.h"
    #include  "libc.h"
    void LibMeshLine(int x, int y, int xsize, int ysize)
```

[Arguments]

int x	:IN	Start coordinate - Horizontal
int y	:IN	Start coordinate - Vertical
int xsize	:IN	Dotted line size - width
int ysize	:IN	Dotted line size - height

[Return values] None

[Description] Draws a dotted line.
This is used to draw a dotted line with a size specified by "xsize" and "ysize" from the start coordinate specified by (x, y).

[Examples of usage]

```
    LibLine(10, 227, 304, 1);
    LibMeshLine(10,228, 306, 1);
    LibMeshLine(10,229, 306, 1);
    LibLine(313, 48, 1, 180);
    LibMeshLine(314, 48, 1, 182);
    LibMeshLine(315, 48, 1, 182);
    LibPutDisp();
```

- Display functions -

[Function name] LibLineClr

[Syntax]

```
#include "define.h"
#include "libc.h"
void LibLineClr(int x, int y, int xsize, int ysize)
```

[Arguments]

```
int x          :IN    Start coordinate - Horizontal
int y          :IN    Start coordinate - Vertical
int xsize      :IN    Size to clear - width
int ysize      :IN    Size to clear - height
```

[Return values] None

[Description] Clears a border.

This is used to clear a border with a size specified by "xsize" and "ysize" from the start coordinate specified by (x, y).

[Examples of usage]

```
LibLine(2, 235, 318, 1);        /* Draws a border. */
LibLine(319, 40, 1, 196);       /* Draws a border. */
LibLineClr(2, 235, 100, 1);     /* Partial clear */
LibLineClr(319, 40, 1, 100);    /* Partial clear */
LibPutDisp();
```

- Display functions -

[Function name] LibLineCplmnt

[Syntax]

```
#include "define.h"
#include "libc.h"
void LibLineCplmnt(int x, int y, int xsize, int ysize)
```

[Arguments]

```
int x          :IN   Start coordinate - Horizontal (Graphic system)
int y          :IN   Start coordinate - Vertical (Graphic system)
int xsize      :IN   Border size - width
int ysize      :IN   Border size - height
```

[Return values] None

[Description] Draws a border. (Bit inversion.)
This is used to draw a border using the bit pattern inversion with a size specified by "xsize" and "ysize" from the start coordinate specified by (x, y). The VRAM data before drawing is inverted.

[Note] This function does not perform data transfer to D/D. Therefore, newly set data is not displayed actually (invalid) unless LibPutDisp is executed.

[Examples of usage]

```
LibLineCplmnt(2, 235, 318, 1);  /* Cropping border */
LibLineCplmnt(319, 40, 1, 196); /* Cropping border */
LibPutDisp();
```

- Display functions -

[Function name] `LibDotOn`

[Syntax]

```
    #include  "define.h"
    #include  "libc.h"
    void LibDotOn(int x, int y)
```

[Arguments]

```
    int x     :IN       Coordinate - Horizontal
    int y     :IN       Coordinate - Vertical
```

[Return values] `None`

[Description] Draws a dot.

This is used to draw a dot with a size of one dot both for the width and height at the coordinate specified by (x, y).

[Examples of usage]

```
        int     cnt,pos;

        pos = 0;
        for (cnt = 0; cnt < 91; cnt++) {
            LibDotOn(71 + pos, 162);
            pos += 2;
        }
        LibPutDisp();
```

- Display functions -

[Function name] `LibDotOff`

[Syntax]

```
#include "define.h"
#include "libc.h"
void LibDotOff(int x, int y);
```

[Arguments]

```
int x     :IN      Coordinate - Horizontal (Graphic system)
int y     :IN      Coordinate - Vertical (Graphic system)
```

[Return values] `None`

[Description] Clears a dot.

This is used to clear a dot with a size of one dot both for the width and height at the coordinate specified by (x, y).

[Examples of usage]

```
#define FIELD_DATA_X    30  /* Data display: start X coordinate
                                 in the input field */
#define FIELD_CURS_Y    31  /* Cursor display: Y coordinate
                                 in the input field   */

int     dsp_x,dot

for(; dot != 0; dot--){
    LibDotOff(FIELD_DATA_X+dsp_x+dot, FIELD_CURS_Y);
}
LibPutDisp();
```

- Display functions -

[Function name] `LibPutProFont`

[Syntax]

```
#include "define.h"
#include "libc.h"
int LibPutProFont(byte type, byte code, int x_pos,int y_pos)
```

[Arguments]

```
byte     type                :IN  Display font data type
                   IB_PFONT1 :Normal
                   IB_PFONT2 :Bold
                   IB_PFONT3 :For title
                   IB_CG57FONT:  5*7
byte     code                :IN  Character code
int      x_pos               :IN  Coordinate - Horizontal
int      y_pos               :IN  Coordinate - Vertical
```

[Return values] `Next abscissas`

[Description] Displays a proportional font/5*7 font.
A character specified by the "code" and "type" (font type) is displayed. A horizontal coordinate calculated by adding the character size specified in "code" to the horizontal coordinate specified by "x_pos" is returned as a next display coordinate.

[Examples of usage]

```
LibPutProFont(IB_PFONT2,'-',50,30); /*  Display negative sign.  */
```

- Display functions -

[Function name] `LibGetProFontSize`

[Syntax]

```
    #include  "define.h"
    #include  "libc.h"
    int LibGetProFontSize(byte type, byte code)
```

[Arguments]

```
    byte     type               :IN  Display font data type
                      IB_PFONT1 : Normal
                      IB_PFONT2 : Bold
                      IB_PFONT3 : For title
    byte     code               :IN  Character code
```

[Return values] `A character size.`

[Description] Gets the width of the proportional font.

- Display functions -

[Function name] `LibPutProStr`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
void LibPutProStr(byte type,int x,int y,byte *string,int x_size)
```

[Arguments]
```
byte      type                :IN  Display font data type
                   IB_PFONT1      : Normal
                   IB_PFONT2      : Bold
                   IB_PFONT3      : For title
                   IB_CG57FONT    : 5*7
int       x                   :IN  Coordinate - Horizontal
int       y                   :IN   Coordinate - Vertical
byte      *string             :IN  Character string
int       x_size              :IN  Area (Horizontal)
```

[Return values] `None`

[Description] Displays a proportional font character string.
A character string specified in "string" is displayed with a font specified in "type". Data exceeding "x_size" is not displayed.

[Examples of usage]
```
        LibPutProStr(IB_PFONT1,5,11+i*9,"ABCDEFGHIJKLMN",70);
```

- Display functions -

[Function name] `LibGetProStrSize`

[Syntax]
```
#include "define.h"
#include "libc.h"
int LibGetProStrSize(byte type,byte *string)
```

[Arguments]
```
byte      type              : IN      Display font data type
                   IB_PFONT1 : Normal
                   IB_PFONT2 : Bold
                   IB_PFONT3 : For title
                   IB_CG57FONT:  5*7
byte      *string           : IN  Character string
```

[Return values] `Dot length of string`

[Description] Gets the size of a proportional font character string.
This function returns a dot length when a character string specified in “string” is displayed with a font specified in “type”.

- Display functions -

[Function name] `LibPut35Font`

[Syntax]

```
#include  "define.h"
#include  "libc.h"
void LibPut35Font(byte code,int x_pos,int y_pos)
```

[Arguments]

byte	code	:IN	Character code
int	x_pos	:IN	Coordinate - Horizontal
int	y_pos	:IN	Coordinate - Vertical

[Return values] None

[Description] Displays a 3 * 5 font.
Displays a character code specified in "code" at specified coordinate.

[Note] However, only the following character codes can be used. If other codes are used, they are painted.

```
0 1 2 3 4 5 6 7 8 9 A M P . , 0x20 0x2d 0x3a 0x2f 0x7e
```

- Display functions -

[Function name]   `LibPut35Str`

[Syntax]

```
    #include  "define.h"
    #include  "libc.h"
    void LibPut35Str(int x,int y,byte *string)
```

[Arguments]

```
    int      x                 :IN     Coordinate - Horizontal
    int      y                 :IN     Coordinate - Vertical
    byte    *string    :IN     Character string
```

[Return values]   None

[Description] Displays a 3 * 5 font character string.

[Note]   However, only the following character codes can be used. If other codes are used, they are filled.

```
0 1 2 3 4 5 6 7 8 9 A M P . , 0x20 0x2d 0x3a 0x2f 0x7e
```

- Display functions -

[Function name] `LibReverse`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibReverse(int x, int y, int xsize, int ysize)
```

[Arguments]

```
int x              :IN     Start coordinate - Horizontal
int y              :IN     Start coordinate - Vertical
int xsize          :IN     Reverse area   Width
int ysize          :IN     Reverse area   Height
```

[Return values] `None`

[Description] Reverses a rectangular area.
This is used to reverse the area with a size specified by "xsize" and "ysize" from the start coordinate specified by (x, y).

[Examples of usage]

```
LibReverse( 76,  1, 33, 14);    /* Invert */
```

- Display functions -

[Function name] `LibMesh`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibMesh(int px, int py, int xsize, int ysize)
```

[Arguments]

```
int px            :IN     Start coordinate - Horizontal
int py            :IN     Start coordinate - Vertical
int xsize         :IN     Reverse area   Width
int ysize         :IN     Reverse area   Height
```

[Return values] `None`

[Description] Shades a rectangular area.

This is used to shade the area with a size specified by "xsize" and "ysize" from the start coordinate specified by (x, y).

[Examples of usage]

```
LibMesh( 76,  1, 33, 14);   /* Shade */
```

- Display functions -

[Function name] `LibBox`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibBox(int x, int y, int xsize, int ysize, byte type)
```

[Arguments]

`int x`	`:IN`	`Coordinate - Horizontal`
`int y`	`:IN`	`Coordinate - Vertical`
`int xsize`	`:IN`	`Box size  Width`
`int ysize`	`:IN`	`Box size  Height`
`byte type`	`:IN`	`Box Line  Width`

[Return values] `None`

[Description] Draws a box.

This is used to draw a box with a size specified by "xsize" and "ysize" from the start coordinate specified by (x, y).

- Display functions -

[Function name] `LibPutGraph`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibPutGraph(int x, int y, const byte far *graph)
```

[Arguments]

```
int x               :IN     Coordinate - Horizontal
int y               :IN     Coordinate - Vertical
byte far *graph     :IN     Graphic data
```

[Return values] `None`

[Description] Displays graphic data. Displays only graphic data defined by “graph”.

[Examples of usage]

```
static byte far GraphCenter[] = /* Center mark difinition */
{
    GSIZE(11, 11),
    0x04,0x00,0x0A,0x00,0x1D,0x00,
    0x3A,0x80,0x7D,0x40,0xAA,0xA0,
    0x57,0xC0,0x2B,0x80,0x17,0x00,
    0x0A,0x00,0x04,0x00
};

int main(void){
    LibPutGraph( 183, 12, GraphCenter );  /* Center mark display */
    LibPutDisp();
}
```

- Display functions -

[Function name] `LibPutGraphM`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibPutGraphM(int x, int y, const byte far *graph)
```

[Arguments]

```
int x              :IN     Coordinate - Horizontal
int y              :IN     Coordinate - Vertical
byte far *graph    :IN     Graphic data
```

[Return values] `None`

[Description] Shades and displays graphic data.
Shades and displays graphic data already defined by "graph".

- Display functions -

[Function name] `LibPutGraphO`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     #include    "l_define.h"
     void LibPutGraphO(int px, int py, byte far *ptn, int mode)
```

[Arguments]

```
     int       px                  :IN  Coordinate - Horizontal
     int       py                  :IN  Coordinate - Vertical
     byte far  *ptn                :IN  Graphic data
     int       mode                :IN  Write mode
                                   IB_GPOVER: Overwrite
                                   IB_GPOR: OR
                                   IB_GPAND: AND
                                   IB_GPREV: Reverse
                                   IB_GPMESH: Shade
```

[Return values] `None`

[Description] Displays graphic data with the write mode specification.
Displays graphic data defined by "graph" in the way corresponding to each write mode.

- Display functions -

[Function name] `LibPutFarData`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
void LibPutFarData(int x, int y, int no)
```

[Arguments]

int	x	:IN	Coordinate - Horizontal
int	y	:IN	Coordinate - Vertical
int	no	:IN	Graphic data number

[Return values] None

[Description] Displays built-in graphic data. System built-in graphic data corresponding to a number specified by "no" is displayed.

[Examples of usage]

```
/* Icon display */
LibPutFarData( 44,  0, 11);     /*  12 * 12 */
LibPutFarData( 57,  0, 59);     /*  25 * 12 */
LibPutFarData( 83,  0, 31);     /*  14 * 12 */
LibPutFarData( 98,  0, 36);     /*  30 * 12 */
```

- Display functions -

[Function name] `LibGetGraph`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    void LibGetGraph(int x, int y, int xsize, int ysize, byte far *ubfptr)
```

[Arguments]

```
    int x              :IN      Start coordinate - Horizontal
    int y              :IN      Start coordinate - Vertical
    int xsize          :IN      Read area - Width
    int ysize          :IN      Read area - Height
    byte far *ubfptr   :OUT     Read buffer
```

[Return values] `None`

[Description] Gets data for the rectangular area from the VRAM.
This function gets VRAM data for a size specified by "xsize" and "ysize" starting from the coordinate specified by (x, y). The data format gotten by this function is the same as that specified by LibPutGraph family.
This function makes it possible to save data.

[Examples of usage]

```
    byte    kind,
            ibuf[1024], wbuf[1024];

    kind    = icon->kind;
    xsize   = icon->tch->x2 - icon->tch->x1 + 1;
    ysize   = icon->tch->y2 - icon->tch->y1 + 1;

    *(word *)ibuf       = xsize;
    *(word *)(ibuf+2)   = ysize;
    LibGetGraph(x, y, xsize, ysize, &ibuf[4]);

    LibIconMoveUp(ibuf, wbuf, kind);
    LibPutGraph(x, y, wbuf);
```

- Display functions -

[Function name] `LibGrpUp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGrpUp(int x, int y, int xsize, int ysize, int up_size)
```

[Arguments]

```
int      x         :IN     Coordinate - Horizontal
int      y         :IN     Coordinate - Vertical
int      xsize     :IN     Size - Width
int      ysize     :IN     Size - Height
int      up_size   :IN     Scroll up size
```

[Return values] `None`

[Description] Scrolls up the rectangular area.
This function scrolls up a specified coordinate and a size with an amount specified in "up_size".

- Display functions -

[Function name] `LibGrpDwn`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGrpDwn(int x, int y, int xsize, int ysize, int dw_size)
```

[Arguments]

```
int     x                  :IN     Coordinate - Horizontal
int     y                  :IN     Coordinate - Vertical
int     xsize     :IN     Size - Width
int     ysize     :IN     Size - Height
int     dw_size   :IN     Scroll down size
```

[Return values] `None`

[Description] Scrolls down the rectangular area.
This function scrolls down a specified coordinate and a size with an amount specified in "dw_size".

- Display functions -

[Function name] `LibPutDisp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibPutDisp(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Transfers VRAM data to D/D. (Entire screen)

- Display functions -

[Function name] `LibPutDispBox`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibPutDispBox(int x, int y, int xsize, int ysize)
```

[Arguments]

```
int x     :IN     Start coordinate - Horizontal
int y     :IN     Start coordinate - Vertical
int xsize          :IN     Area size  Width
int ysize          :IN     Area size  Height
```

[Return values] `None`

[Description] Transfers VRAM data to D/D. (Area specification.)
This is used to transfer the rectangular area with a size specified by "xsize" and "ysize" from the start coordinate specified by (x, y) to D/D.

[Note] This performs the same operation with the entire screen transfer LibPutDisp() on the simulator.

[Examples of usage]

```
LibIconPrintR(icon);    /* Icon highlight */
LibPutDispBox(x, y, xsize, ysize);  /* Display valid     */
```

- Display functions -

[Function name] `LibClrDisp`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibClrDisp(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Clears the entire screen.

- Display functions -

[Function name] `LibClrBox`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibClrBox(int x, int y, int xsize, int ysize)
```

[Arguments]

```
int x            :IN     Start coordinate - Horizontal
int y            :IN     Start coordinate - Vertical
int xsize        :IN     Size to clear - width
int ysize        :IN     Size to clear - height
```

[Return values] `None`

[Description] Clears the rectangular area.

This is used to clear the rectangular area with a size specified by "xsize" and "ysize" from the start coordinate specified by (x, y).

- Display functions -

[Function name] `LibSetDispMode`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibSetDispMode(bool flag)
```

[Arguments]

```
bool flag                  :IN     Write mode
                                FALSE: Not transfer to D/D.
                                TRUE: Transfer to D/D
```

[Return values] `None`

[Description] Sets the write mode to D/D.
The real time transfer is performed when setting this to TRUE. That is, this function makes it possible to display a drawing content instantaneously without executing LibPutDisp().
The initial value is FALSE.

[Note] The performance of the complete display is down when this is set to TRUE. Therefore, it is recommended to limit a use of this function.

- Display functions -

[Function name] `LibInitDisp`

[Syntax]
```
     #include    "define.h"
     #include    "libc.h"
     void LibInitDisp(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Initializes the origin of the LCD text coordinate.

[Note] This system does not use data for the text coordinate system. Thus it is not necessary to execute this function.

- Display functions -

[Function name] `LibSKeyRevSub`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibSKeyRevSub(int x, int y, int x2, int y2, byte rev,byte type)
```

[Arguments]

```
int      x                  :IN     Top left abscissa
int      y                  :IN     Top left ordinate
int      x2                 :IN     Bottom right abscissa
int      y2                 :IN     Bottom right ordinate
byte     rev                :IN     Highlighted (reverse) pattern
                 IB_GDS_KEYREV: Highlighted (reverse) rectangle.
                 IB_GDS_KEYREVR: Undo reverses a rectangular.
byte     type               :IN     Patterns for format
                 IB_GDS_KREVP1: Pattern 1, normal reverse
                 IB_GDS_KREVP2: Pattern 2, bottom right shadow
                 IB_GDS_KREVP3: Pattern 3, border & bottom right shadow
                 IB_GDS_KREVP4: Pattern 4, border
```

[Return values] None

[Description] Provides the reverse appearance to the rectangular area. This function performs data transfer to D/D. Therefore, it is not necessary to issue LibPutDisp().

- Display functions -

[Function name] `LibGdsBox`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGdsBox(int x,int y,int x2,int y2)
```

[Arguments]

```
int      x                  :IN     Top left abscissa
int      y                  :IN     Top left ordinate
int      x2                 :IN     Bottom right abscissa
int      y2                 :IN     Bottom right ordinate
```

[Return values] `None`

[Description] Draws a box by overriding.

- Display functions -

[Function name] `LibGdsBoxMesh`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGdsBoxMesh(int x,int y,int x2,int y2)
```

[Arguments]

```
int      x                 :IN     Top left abscissa
int      y                 :IN     Top left ordinate
int      x2                :IN     Bottom right abscissa
int      y2                :IN     Bottom right ordinate
```

[Return values] `None`

[Description] Draws a box with dotted lines.

- Display functions -

[Function name] `LibGdsBoxClr`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    void LibGdsBoxClr(int x,int y,int x2,int y2)
```

[Arguments]

int	x	:IN	Top left abscissa
int	y	:IN	Top left ordinate
int	x2	:IN	Bottom right abscissa
int	y2	:IN	Bottom right ordinate

[Return values] None

[Description] Clears a box.

- Display functions -

[Function name] `LibGdsBoxCmp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGdsBoxCmp(int x,int y,int x2,int y2)
```

[Arguments]

int	x	:IN	Top left abscissa
int	y	:IN	Top left ordinate
int	x2	:IN	Bottom right abscissa
int	y2	:IN	Bottom right ordinate

[Return values] None

[Description] Draws a box using XOR operator.

- Display functions -

[Function name] `LibGdsClr`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    void LibGdsClr(int x,int y,int x2,int y2)
```

[Arguments]

```
    int      x                 :IN     Top left abscissa
    int      y                 :IN     Top left ordinate
    int      x2                :IN     Bottom right abscissa
    int      y2                :IN     Bottom right ordinate
```

[Return values] `None`

[Description] Clears a rectangular area.

- Display functions -

[Function name] `LibGdsReverse`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    void LibGdsReverse(int x,int y,int x2,int y2)
```

[Arguments]

```
    int     x                  :IN     Top left abscissa
    int     y                  :IN     Top left ordinate
    int     x2                 :IN     Bottom right abscissa
    int     y2                 :IN     Bottom right ordinate
```

[Return values] `None`

[Description] Reverses a rectangular area.

- Display functions -

[Function name] `LibGdsMesh`

[Syntax]
```
    #include    "define.h"
    #include    "libc.h"
    void LibGdsMesh(int x,int y,int x2,int y2)
```

[Arguments]
```
    int      x                  :IN     Top left abscissa
    int      y                  :IN     Top left ordinate
    int      x2                 :IN     Bottom right abscissa
    int      y2                 :IN     Bottom right ordinate
```

[Return values] `None`

[Description] Shades a rectangular area.

- Display functions -

[Function name] `LibGdsDotOn`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGdsDotOn(int x,int y)
```

[Arguments]

```
int       x                   :IN     Top left abscissa
int       y                   :IN     Top left ordinate
```

[Return values] None

[Description] Draws a dot.

- Display functions -

[Function name] `LibGdsDotOff`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibGdsDotOff(int x,int y)
```

[Arguments]

```
     int     x                  :IN     Top left abscissa
     int     y                  :IN     Top left ordinate
```

[Return values] `None`

[Description] Clears a dot.

- Display functions -

[Function name] `LibGdsDotCmp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGdsDotCmp(int x,int y)
```

[Arguments]

```
int      x                  :IN     Top left abscissa
int      y                  :IN     Top left ordinate
```

[Return values] `None`

[Description] Draws a dot using XOR operator.

- Display functions -

[Function name] `LibGdsLine`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGdsLine(int x,int y,int x2,int y2)
```

[Arguments]

```
int      x        :IN     Top left abscissa
int      y        :IN     Top left ordinate
int      x2       :IN     Bottom right abscissa
int      y2       :IN     Bottom right ordinate
```

[Return values] `None`

[Description] Draws a line by overriding.

- Display functions -

[Function name] `LibGdsLineClr`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibGdsLineClr(int x,int y,int x2,int y2)
```

[Arguments]

```
     int      x        :IN     Top left abscissa
     int      y        :IN     Top left ordinate
     int      x2       :IN     Bottom right abscissa
     int      y2       :IN     Bottom right ordinate
```

[Return values] None

[Description] Clears a line.

- Display functions -

[Function name] `LibGdsLineMesh`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGdsLineMesh(int x,int y,int x2,int y2)
```

[Arguments]

```
int      x        :IN      Top left abscissa
int      y        :IN      Top left ordinate
int      x2       :IN      Bottom right abscissa
int      y2       :IN      Bottom right ordinate
```

[Return values] `None`

[Description] Draws a dotted line.

- Display functions -

[Function name] `LibGdsLineCmp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGdsLineCmp(int x,int y,int x2,int y2)
```

[Arguments]

```
int      x        :IN    Top left abscissa
int      y        :IN    Top left ordinate
int      x2       :IN    Bottom right abscissa
int      y2       :IN    Bottom right ordinate
```

[Return values] `None`

[Description] Draws a line using XOR operator.

- Display functions -

[Function name] `LibPutBoxSub`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibPutBoxSub(int x,int y,int x2,int y2,byte type)
```

[Arguments]

```
int      x        :IN     Top left abscissa
int      y        :IN     Top left ordinate
int      x2       :IN     Bottom right abscissa
int      y2       :IN     Bottom right ordinate
byte    type      :IN     Drawing patterns
                            IB_GDS_OVER  Overwrite
                            IB_GDS_AND  Clear
                            IB_GDS_MESH  Dotted line
                            IB_GDS_XOR  XOR draw
```

[Return values] `None`

[Description] Draws a box with the option specification. Draws a box with a type specified in "type" at specified coordinate.

- Display functions -

[Function name] `LibCngeBoxSub`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibCngeBoxSub(int x,int y,int x2,int y2,byte type)
```

[Arguments]

```
int      x        :IN     Top left abscissa
int      y        :IN     Top left ordinate
int      x2       :IN     Bottom right abscissa
int      y2       :IN     Bottom right ordinate
byte    type      :IN     Patterns for format
                            IB_GDS_AND  Clear
                            IB_GDS_REV  Reverse
                            IB_GDS_MESH  Shade
```

[Return values] `None`

[Description] Change the appearance of the rectangular area. Applies a pattern specified in "type" to the rectangular area that locates at specified coordinate.

- Display functions -

[Function name] `LibPutDotSub`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     #include    "l_define.h"
     #include    "l_libc.h"
     void LibPutDotSub(int x,int y,byte type)
```

[Arguments]

```
     int       x         :IN     Top left abscissa
     int       y         :IN     Top left ordinate
     byte      type      :IN     Drawing patterns
                                   IB_GDS_OR   Turns ON
                                   IB_GDS_AND  Turns OFF
                                   IB_GDS_XOR  XOR
```

[Return values] `None`

[Description] Draws a dot with the drawing pattern specification. Draws a dot with a type specified in "type" at specified coordinate.

- Display functions -

[Function name] `LibPutLineSub`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibPutLineSub(int x,int y,int x2,int y2,byte type)
```

[Arguments]

```
int      x        :IN     Start point abscissa
int      y        :IN     Start point ordinate
int      x2       :IN     End point abscissa
int      y2       :IN     End point ordinate
byte     type     :IN     Drawing patterns
                            IB_GDS_OR    Solid line
                            IB_GDS_AND   Clear
                            IB_GDS_MESH   Dotted line
                            IB_GDS_XOR   XOR
                            IB_GDS_XOR   XOR writing
```

[Return values] `None`

[Description] Draws a line with the drawing pattern specification. Draws a line with a type specified in "type" at specified coordinate.

- Display functions -

[Function name] `LibGrphUpSideDown`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    void LibGrphUpSideDown(byte *gw, const byte *gs)
```

[Arguments]

```
    byte              *gw      :OUT  Write buffer after flipping.
    const byte        *gs      :IN   Flipping source buffer.
```

[Return values] `None`

[Description] Flips graphic data specified by "gs" vertically, and write the data to "gw".

- Display functions -

[Function name] `LibStringDsp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibStringDsp(byte *strg, word xps, word yps, word lmtx, byte font);
```

[Arguments]

```
byte     *strg    :IN    Character string buffer
word     xps      :IN    Display start position - Horizontal
word     yps      :IN    Display start position - Vertical
word     lmtx     :IN    Display end position - Horizontal
byte     font     :IN    Font type
```

[Return values] `None`

[Description] Displays a character string specified in “strg[]” at specified coordinates. If a character string exceeds the end coordinate, “…” is put at the end of the character string displayed. This function is used to display a character string meeting the common specification, such as displaying of lists.

- Window functions -

[Function name] `LibOpenWindow`

[Syntax]
```
     #include    "define.h"
     #include    "libc.h"
     bool LibOpenWindow(int x, int y, int xsize, int ysize)
```

[Arguments]
```
     int x             :IN     Coordinate - Horizontal
     int y             :IN     Coordinate - Vertical
     int xsize         :IN     Size - Width
     int ysize         :IN     Size - Height
```

[Return values]
```
Result   TRUE: Succeeded
         TRUE: Failed
```

[Description] Opens a window. Opens a window with specified size at specified coordinate.

- Window functions -

[Function name] `LibOpenWindowS`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibOpenWindowS(byte flame, word px, word py, word xsize, word ysize)
```

[Arguments]

```
byte      flame              :IN     Frame option
                                 OFF  : No border
                                 ON   : With border
word      px                 :IN     Coordinate - Horizontal
word      py                 :IN     Coordinate - Vertical
word      xsize              :IN     Border size - width
word      ysize              :IN     Border size - height
```

[Return values] `None`

[Description] Opens a window with the option specification.
Opens a window with the specified size and the frame option at specified coordinate.

[Note] Some of the display system functions require to consider that the origin (0, 0) of the coordinates is upper-left of the window opened by this function.
Those functions are listed below:

```
LibLine()
LibMeshLine()
LibGrpUp()
LibGrpDwn()
LibPutDispBox()
LibClrBox()
LibBox()
```

- Window functions -

[Function name] `LibCloseWindow`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCloseWindow(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Closes a window. Closes the last-opened window and restores the original VRAM data.

[Note] It causes an error if attempting to close more windows than actually opened.

- Touch functions -

[Function name] `LibTchInit`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibTchInit(void)
```

[Arguments] None

[Return values] None

[Description] Initializes the touch information.

[Examples of usage]

```
LibTchInit();    /* Initialization       */

for(;;)
{
    LibTchWait(&tsts);    /* Wait for a touch information */
    switch(tsts.obj){
        •
        •
        •
```

- Touch functions -

[Function name] `LibTchStackClr`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibTchStackClr(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Clears the stack contents of the touch information table. This function discards all of the touch information registered (stacked) by "LibTchStackPush()".

[Examples of usage]

```
LibTchStackClr();  /* Clear stack contents */

LibTchStackPush(NULL);
LibTchStackPush(TchHardIcon);  /* Hardware icon */
LibTchStackPush(TchSearchInput);    /* Dragging area */
LibTchStackPush(TchSearch);    /* Various search buttons   */
LibTchStackPush(TchMenu);    /* Menu */
        •
        •
        •
```

- Touch functions -

[Function name] `LibTchStackPop`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
TCHTBL far *LibTchStackPop(void)
```

[Arguments] `None`

[Return values]

```
Result   NULL     : Stack under flow
         Non-NULL : Touch information table address
```

[Description] Gets the touch information table registered last (discards). This function pops up the touch information table stacked by “LibTchStackPush()” and returns that address.

[Examples of usage]

```
/* Clear Screen definition table */
LibTchStackPop();
LibTchStackPop();
LibTchStackPop();
LibTchStackPop();
```

- Touch functions -

[Function name] `LibTchStackPush`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
bool LibTchStackPush(TCHTBL far *tbl)
```

[Arguments]
```
TCHTBL far *tbl    :IN     Touch information table
```

[Return values]
```
Result    FALSE     : Stack over flow
          TRUE      : Normal
```

[Description] Registers the touch information table.
This function stacks the touch information table address specified by "tbl". If it is successful, the function returns TRUE.

[Note] Insert the termination identification data to the end of the touch information table. It is recommended to register "0" as the argument before registering the first table in order to avoid malfunctions.

[Examples of usage]
```
#define OBJ_HIC_KEY01       0xc010      /* Key1*/

TCHTBL far TchData[] =
{
    13, 196, 52,224,    /* Key button definition */
    ACT_MAKE | ACT_MOVE_IN |ACT_MOVE_OUT | ACT_BREAK_IN,
    OBJ_HIC_KEY01,
    0x0000,

      0,  0,  0,  0,    /* End recognition */
    ACT_NONE,
    OBJ_END,
    0x0000
};

T_ICON far HanKey =  /* AC */
{
    &TchData[0], NULL, NULL,0x01
};

LibTchStackClr();       /* Clear stack contents */

LibTchStackPush(NULL);
LibTchStackPush(TchHardIcon);       /* Hardware icon */
LibTchStackPush(TchData);   /* AC key button registration */
```

- Touch functions -

[Function name] `LibTchHardIcon`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    #include    "l_define.h"
    #include    "l_libc.h"
    void LibTchHardIcon(TCHSTS *tsts,byte opt)
```

[Arguments]

```
    TCHSTS    *tsts     :IN     Touch status information
    byte      opt       :IN     Handwriting mode call option
                                  0:   Call
                                  1:   No Call
```

[Return values] `None`

[Description] Controls touching of the hardware icon. This can be called during common wait for touching.

- Touch functions -

[Function name] `LibTchWait`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibTchWait(TCHSTS *tsts)
```

[Arguments]

```
const TCHSTS *tsts          :OUT    Touch status information
```

[Return values] `None`

[Description] Gets the touch information table. This function always monitors the pen-touch. If a touch occurs, the function updates the status information of “tsts”. An effective touch area is specified by “LibTchStackPush()”.

The touch table which is prepared at the library.

TchHardIcon	:for HardIcon
TchActionKey	:for ActionControl key
TchAllDsp	:for AllDisplay

[Note] Waits until the valid action occurs.

[Examples of usage]

```
LibTchInit();

while(1){

    LibTchWait(&tsts);

    switch(tsts.obj){
        case OBJ_HIC_FUNC:
        break;
        case OBJ_IC_DUAL:
        •
        •
        •
```

- Touch functions -

[Function name] `LibTchWaitScan`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibTchWaitScan(TCHSTS *tsts)
```

[Arguments]

```
TCHSTS *tsts                :OUT    Touch status information
```

[Return values] `bool` : `TRUE: Input`
`FALSE: No input`

[Description] Check if a touch or key event has occurred in the input buffer and, if there is an event, output that event to touch status information.
When the buffer is empty, FALSE is returned as the return value, and the touch status information is not updated.
When a keystroke occurs from the external keyboard, only its ESC key is output after converted to the hardware icon ESC key. Any stroke of external key other than that is NOP.
If you want to assign this an individual operation for an external key, judge that with the combination of LibTchWaitSKey() and LibGetOptKeyCode().

[Note] Because no event is output if the buffer is empty even during a touch or keypress, this cannot be used for a judgment during a touch or keypress. The caller must control the state.
Because information of the last action is not updated by this function, the repeat-off function will not operate if a repeat action is output from this function.

[Examples of usage]

```
TCHSTS tsts;
bool exist;
int state = STATE_IDOL;
exist = LibTchWaitScan(&tsts);
if (exist) {
  switch (tsts.act) {
    case ACT_MAKE :
    case ACT_DOWN :
          /* Processing with input */
         state = STATE_INPUT;            // State with input
         break;
    case ACT_BREAK :
         state = STATE_IDOL;             // State without input
         break;
  }
}
```

```
else {
   if (state==STATE_IDOL) {
      /* Processing without input */
     }
 }
```

- Touch functions -

[Function name] `LibIconPrint`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibIconPrint(const T_ICON far *icon)
```

[Arguments]

```
const T_ICON far *icon      :IN     Icon information
```

[Return values] `None`

[Description] Displays an icon. This function displays an icon based on the icon information specified by "icon".

[Note] This function does not perform data transfer to D/D. Therefore, newly set data is not displayed actually (invalid) unless LibPutDisp is executed.

[Examples of usage]

```
#define OBJ_IC_01   0x8107
#define OBJ_IC_02   0x8108

byte far Graph01[] =    /* Icon data1*/
{
    GSIZE(28, 16),
    0xFF,0xFF,0xFF,0xE0,0x80,0x00,0x00,0x30,0x80,0x00,0x00,0x30,0x80,
    0x42,0x1F,0x30,0x84,0x24,0x11,0x30,0x8D,0xFF,0x1F,0x30,0x9C,0xA2,
    0x11,0x30,0xBC,0xEA,0x1F,0x30,0x9C,0xAA,0x11,0x30,0x8C,0xEA,0x11,
    0x30,0x84,0xA2,0x21,0x30,0x80,0xA6,0x43,0x30,0x80,0x00,0x00,0x30,
    0x80,0x00,0x00,0x30,0xFF,0xFF,0xFF,0xF0,0x7F,0xFF,0xFF,0xF0
};

byte far Graph02[] =    /* Icon data2*/
{
    GSIZE(28, 16),
    0xFF,0xFF,0xFF,0xE0,0x80,0x00,0x00,0x30,0x80,0x00,0x00,0x30,0x9E,
    0xF1,0xF0,0x30,0x8A,0x51,0x14,0x30,0x86,0x31,0xF6,0x30,0x9A,0xD1,
    0x17,0x30,0x83,0x11,0xF7,0xB0,0x9F,0xF1,0x17,0x30,0x84,0x41,0x16,
    0x30,0x82,0x82,0x14,0x30,0x9F,0xF4,0x30,0x30,0x80,0x00,0x00,0x30,
    0x80,0x00,0x00,0x30,0xFF,0xFF,0xFF,0xF0,0x7F,0xFF,0xFF,0xF0
};

static  TCHTBL far TchSubMenuBar[] = {
    146, 21,173, 36,    ACT_ICON,   OBJ_IC_01,  0x0000, /* Icon1 */
    176, 21,203, 36,    ACT_ICON,   OBJ_IC_02,  0x0000, /* Icon2 */
};

static T_ICON far Ticon01= {&TchSubMenuBar[0],Graph01,NULL,0x00};
static T_ICON far Ticon02= {&TchSubMenuBar[1],Graph02,NULL,0x00};

LibIconPrint(&Ticon01); /* Icon1 display*/
LibIconPrint(&Ticon02); /* Icon2 display*/

LibPutDisp();
```

[Supplement] **Configurations of `T_ICON` structure**

`TCHTBL far *tch;` Icon touch table

Only x1, y1, x2, and y2 are used.

This provides the information on icon display position and size.

`byte far *ngp;` Graphic pattern for normal icon display

When NULL is put, this enters an unspecified status.

If this is not specified, a pattern written in the VRAM is used.

`byte far *rgp;` Graphic pattern for reverse display of icon

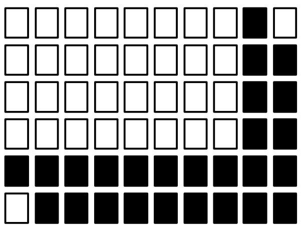
When NULL is input, this enters unspecified status.

If this is not specified, a pattern is created from the normal pattern. If the normal pattern is also not specified, a pattern is created from that written in the VRAM.

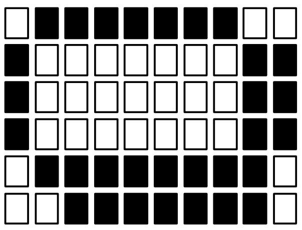
`byte kind;` Icon type (See the figures below.)

This is ignored when the highlighted (reverse) display pattern is specified.

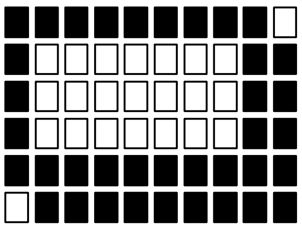
0x00:



0x01:



0x02:



0x03:

Normal highlighted rectangle

- Touch functions -

[Function name] `LibIconPrintR`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
void LibIconPrintR(const T_ICON far *icon)
```

[Arguments]
```
const T_ICON far *icon       :IN     Icon information
```

[Return values] None

[Description] Performs the reverse display of the icon. Displays the reverse icon based on the icon information specified by "icon", or reverses the specified rectangular area. When NULL is assigned to both "icon->ngp" (normal graphic pattern) and "icon->ngp" (reverse graphic pattern), the current VRAM data is inverted.

[Note] This function does not perform data transfer to D/D. Therefore, newly set data is not displayed actually (invalid) unless LibPutDisp() is executed.

[Examples of usage]
```
#define OBJ_HIC_KEY01       0xc010      /* Key1*/

TCHTBL far TchData[] =
{
    13, 196, 52,224,    /* Key button definition */
    ACT_MAKE | ACT_MOVE_IN |ACT_MOVE_OUT | ACT_BREAK_IN,
    OBJ_HIC_KEY01,
    0x0000,

      0,  0,  0,  0,    /* End recognition  */
    ACT_NONE,
    OBJ_END,
    0x0000
};

T_ICON far HanKey =
{
    &TchData[0], NULL, NULL,0x01
};

TCHSTS  tsts;

LibTchStackClr();
```

```
LibTchStackPush(NULL);
LibTchStackPush(TchData);

LibTchInit();
for(;;){
  LibTchWait(&tsts);      /* Wait for touch information*/
   switch(tsts.obj){

   case OBJ_HIC_KEY01: /*Blink a button at a glance when a key is pressed.*/

   LibIconPrintR(&HanKey); /* Reverse */
   LibPutDisp();

   LibIconPrint( &HanKey); /* Undo */
   LibPutDisp();

   LibTchInit();
   break;

        default:
   break;
    }
}
```

- Touch functions -

[Function name] `LibIconPrintM`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibIconPrintM(const T_ICON far *icon)
```

[Arguments]

```
const T_ICON far *icon      :IN     Icon information
```

[Return values] None

[Description] Displays an icon with shading. Displays the shaded icon based on the icon information specified by "icon".

[Note] The current VRAM data is not shaded. Be sure to set "icon->ngp" (normal graphic pattern). As touching is not invalid, so it is necessary to invalid it in your application. This function does not perform data transfer to D/D. Therefore, newly set data is not displayed actually (invalid) unless LibPutDisp() is executed.

[Examples of usage]

```
case 0x0000:
    LibIconPrint(   &TiconData[i]   );  /* Normal display   */
    LibPutDisp();
    break;

case 0x0001:
    LibIconPrintR(  &TiconData[i]   );  /* Reverse display   */
    LibPutDisp();
    break;

case 0x0002:
    LibIconPrintM(  &TiconData[i]   );  /* Shaded display */
    LibPutDisp();
    break;
```

- Touch functions -

[Function name] `LibIconClick`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
bool LibIconClick(const T_ICON far *icon, const TCHSTS *tsts)
```

[Arguments]
```
const T_ICON far *icon       :IN     Icon information
const TCHSTS *tsts           :IN     Touch status information
```

[Return values]
```
Icon touch                    TRUE    : Touched.
                              FALSE   : Not touched.
```

[Description] Controls touching of the icon information. This function determines touching of the icon information specified by “icon”. If a touch occurs, the function highlights (blinks) the icon instantaneously and returns TRUE. The highlighted display may depend on the style specified by “icon->kind”. Execution of LibPutDisp() is not necessary since data is transferred (locally) to D/D internally.

[Note] A timing to become TRUE is immediately after the ACT_BREAK_IN action. ACT_MOVE_OUT will not bring TRUE. Thus, it is after generating two actions (at least).
(ACT_MAKE and ACT_BREAK_IN)

[Examples of usage]
```
#define OBJ_HIC_KEY01       0xc010      /* Key1*/

TCHTBL far TchData[] =
{
    13, 196, 52,224,    /* Key button definition */
    ACT_MAKE | ACT_MOVE_IN |ACT_MOVE_OUT | ACT_BREAK_IN,
    OBJ_HIC_KEY01,
    0x0000,

      0,  0,  0,  0,    /* End recognition     */
    ACT_NONE,
    OBJ_END,
    0x0000
};

T_ICON far HanKey =
{
    &TchData[0], NULL, NULL,0x01
};
```

```
TCHSTS  tsts;

LibTchStackClr();
LibTchStackPush(NULL);
LibTchStackPush(TchData);

LibTchInit();
for(;;){
    LibTchWait(&tsts);          /* Wait for touch information */
    switch(tsts.obj){

        case OBJ_HIC_KEY01:
   if (LibIconClick(&HanKey, &tsts) == TRUE){
        •
        •
        •

   }
   break;

        default:
   break;
    }
}
```

- Touch functions -

[Function name] `LibIconClick2`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibIconClick2(const T_ICON far *icon, const TCHSTS *tsts)
```

[Arguments]

```
const T_ICON far *icon      :IN     Icon information
const TCHSTS *tsts          :IN     Touch status information
```

[Return values]

```
Icon touch                TRUE     : Touched.
                          FALSE    : Not touched.
```

[Description] Controls touching of the icon information. This function determines touching of the icon information specified by “icon”. If a touch occurs, the function highlights the icon instantaneously and returns TRUE. The reverse of the icon size is performed without referring a style specified in "icon->kind". This is used for clicking a non-shadow icon. The operation is the same with LibIconClick().

- Touch functions -

[Function name] `LibScrollPrint`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibScrollPrint(T_SCR_POS scr)
```

[Arguments]

```
T_SCR_POS    scr            :in   Scroll bar position information
```

[Return values] `None`

[Description] Displays the scroll bar. This function displays the scroll bar from a position specified by “scr”. It is not necessary to execute functions, such as LibPutDisp(), since data is transferred to D/D internally.

[Supplement] The position information of the scroll bar is as follow:

```
typedef struct T_SCR_POS{
    int     x;      /* Start position (X) of the bar display */
    int     y;      /* Start position (Y) of the bar display */
    int     size;   /* Height of the bar (vertical width)*/
    int     vol;    /* Total number of records.  */
    int     dsp;    /* Number of display records on the screen.*/
    int     pos;    /* Display start data position on the screen. */
} T_SCR_POS;
```

- Touch functions -

[Function name] `LibScrollArrowPrint`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibScrollArrowPrint(T_SCR_POS scr,byte mask)
```

[Arguments]

```
T_SCR_POS     scr           :IN     Scroll bar position information
byte          mask          :IN     Mask information
          SCR_NO_MASK: Displays the up and down arrows without shade.
          SCR_UP_MASK: Shades the up arrow.
          SCR_DWN_MASK: Shades the down arrow.
          SCR_ALL_MASK: Shades both up and down arrows.
```

[Return values] None

[Description] Displays the up and down arrows on the scroll bar. This function displays the scroll bar arrows specified by "scr" according to the shading status specified by "mask".
To make the display valid, it is necessary to execute LibPutDisp() since data is not transferred to D/D internally.

[Note] Only SCR_NO_MASK can display arrows. Others can shade the arrows already displayed. In addition, in LibScrollPrint(), this function is called with specifying SCR_NO_MASK. Therefore the shading must be performed after that.

[Examples of usage]

```
if(ans == 0)){  /* Touches ▲ (arrow) of scroll bar. */
    if(CurPtr>0){
        CurPtr--;   /* Moves the cursor bar one line upward. */
    }
    else{
        if(Scr.pos>0){
  Scr.pos--;      /* Moves the cursor bar one line upward. */
  LibScrollPrint(Scr);
        }
    }

    /* The cursor locates at the top of the first page. */
    if(CurPtr==0 && Scr.pos == 0){
        LibScrollArrowPrint(Scr,SCR_UP_MASK);  /*Arrow ▲ is shaded.*/
        LibPutDisp();
    }
```

- Touch functions -

[Function name] `LibScrollClick`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
int LibScrollClick(const TCHSTS *tsts,T_SCR_POS *scr_pos)
```

[Arguments]

```
const TCHSTS    *tsts    :in     Touch status information
T_SCR_POS   *scr_pos     :in/out Scroll bar position information
```

[Return values]

```
 Touch position    0    Up arrow
                   1    Down arrow
                   2    Intermediate area (display may be changed)
                   4    Scroll BOX is moving.
                   -1   Other touch area
```

[Description] Controls touching of the scroll bar. This function determines a touch position on the scroll bar based on the touch status specified by "tsts", and returns it.
If a portion other than the up/down arrow (display point or non-display point) is touched, the scroll box is moved (including D/D transfer) to calculate the logical data position on the list based on the movement amount, and this data is output to "scr_pos->pos".

[Supplement] When up/down allow is touched, the page break may not be necessarily performed since the allocation of cursor bar movement and others can be expected. Therefore, the bar is not redisplayed within this function.

[Note] The scroll bar position information has to meet to the content of touch area definition.

[Examples of usage]

```
#include    <stdrom.h>
#include    "define.h"
#include    "libc.h"
#include    "ex_com.h"

#define SCR_XPOS        121     /* Horizontal position of scroll bar */
#define SCR__Y          0       /* Vertical position of scroll bar */
#define SCR__SIZE       128     /* Height of scroll bar   */

#define OBJ_SCR_BAR     0xc011

TCHTBL far TchList[] =
{
    /* Scroll bar */
    SCR_XPOS, SCR__Y,SCR_XPOS+SCR_XSIZE-1,SCR__Y+SCR__SIZE-1,
    ACT_SCR_BAR,
    OBJ_SCR_BAR,
    0x0000,

    /* End recognition */
      0,  0,  0,  0,
    ACT_NONE,
    OBJ_END,
    0x0000
};

void main(void){

    T_SCR_POS   Scr;
    TCHSTS      tsts;
    int         ans;

    Scr.x   = SCR_XPOS;    /* Scroll bar position */
    Scr.y   = SCR__Y;   /* Scroll bar position */
    Scr.size= SCR__SIZE;     /* Height of scroll bar */
    Scr.vol = 50;          /* Total number of records */
    Scr.dsp = 10;          /* Number of display records */
    Scr.pos = 0;   /* Start position */

    LibTchStackClr();

    LibTchStackPush(NULL);
    LibTchStackPush(TchList);

    LibScrollPrint(Scr);    /* Scroll bar display */

    LibScrollArrowPrint(Scr,SCR_UP_MASK); /*First, arrow ▲ is shaded.*/

    LibTchInit();
    for(;;){
        LibTchWait(&tsts);
        switch(tsts.obj){
   case OBJ_SCR_BAR:   /* Scroll bar is touched!! */

       scr_pre_chk = LibScrPosCheck(tsts,Scr);

       if(CurPtr==0 && Scr.pos == 0){
           if(scr_pre_chk == 0){
          break;
           }
        }

        if(CurPtr==SCR_DSP-1 && Scr.pos == Scr.vol-Scr.dsp){
            if(scr_pre_chk == 1){
           break;
            }
        }


        ans = LibScrollClick(&tsts,&Scr);

        if      (ans == 0){ /*▲ (up) arrow */
            /* Movement of cursor bar */
        }
        else if (ans == 1){ /* ▼ (down) arrow */
            /* Movement of cursor bar */
        }
        else if (ans == 2){ /* Intermediate area */
            /* List re-dispay     */
        }

        break;

    default:
        break;
          }
      }

  }
```

- Touch functions -

[Function name] `LibScrPosCheck`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
int LibScrPosCheck(TCHSTS tsts,T_SCR_POS scr)
```

[Arguments]

TCHSTS	tsts	:IN	Touch status information
T_SCR_POS	scr	:IN	Scroll bar position information

[Return values]

Touch position	0	▲ (Up) arrow
	1	▼ (Down) arrow
	2	Intermediate area (display may be changed)
	-1	Other touch areas

[Description] Gets a scroll bar touch position. This function determines a touch position on the scroll bar based on the touch status shown in “tsts”, and then returns it.
When making the up/down arrow invalid, it is necessary to check the status using this function.

[Examples of usage]

```
case OBJ_SCR_BAR:   /* Touches the scroll bar. */

    scr_pre_chk = LibScrPosCheck(tsts,Scr); /* Checks in advance. */

    /* The cursor locates at the top of the first page. */
    if(CurPtr==0 && Scr.pos == 0){
       if(scr_pre_chk == 0){  /* ▲ (Up) arrow */
       break;
        }
    }

    /* The cursor locates at the bottom of the last page. */
    if(CurPtr==SCR_DSP-1 && Scr.pos == Scr.vol-Scr.dsp){
        if(scr_pre_chk == 1){  /* ▼ (Down) arrow */
       break;
        }
    }

    ans = LibScrollClick(&tsts,&Scr);
            •
            •
            •
```

- Touch functions -

[Function name] `LibKeyInit`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibKeyInit(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Initializes the generic keyboard. This function must be executed immediately before using the generic keyboard.

- Touch functions -

[Function name] `LibDispKey`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibDispKey(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Displays the generic keyboard.

[Supplement] A keyboard type to be displayed is automatically determined by the value of the system area.

```
extern word far SYS_KEY_FLG;     /* Keyboard     */
extern byte far SYS_KEY_KIND;    /* Keyboard layout types */
```

- Touch functions -

[Function name] `LibGetKeyM`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
byte LibGetKeyM(TCHSTS *tsts)
```

[Arguments]

TCHSTS	*tsts	:OUT	Touch status information

[Return values]

Character code	If a touch area other than the keyboard area is touched, this function returns KEY_NONE.

[Description] Waits for touching of the generic keyboard. This function returns a character code selected on the keyboard. This function also controls changing/displaying the keyboard type when any of the SHIFT, CAPS, CODE, and SMBL keys is pressed.

[Examples of usage]

```
LibKeyInit();       /* Initializes keyboard         */
LibDispKey();       /* Keyboard display (not required onward) */

while(1){
    kycd = LibGetKeyM(&tsts);    /* Wait for a software key. */
    if(kycd==KEY_NONE){          /* Touch outside the keyboard. */
        ●
        ●
        ●
    }
}
```

- Touch functions -

[Function name] `LibCldKeyInit`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCldKeyInit(TCHSTS *tsts, byte *db, int xsp, int ysp, byte type1,
                                                             byte F_Type)
```

[Arguments]

```
TCHSTS   *tsts    :IN    Touch status information
byte     *db      :IN    Date buffer     [8]
int      xsp      :IN    Start point of display    Horizontal
int      ysp      :IN    Start point of display    Vertical
byte     type1    :IN    Display format
                           IN_MODE:    For input
                           DISP_MODE:  For display
                           DISP_YM:    Year-Month
                           DISP_MY:    Month-Year
                           IN_YM:      Year-Month input
byte     F_Type   :IN    Font type
```

[Return values] `None`

[Description] Initializes the calendar keyboard.

- Touch functions -

[Function name] `LibGetCale`

[Syntax]

```
  #include    "define.h"
  #include    "libc.h"
  byte LibGetCale(byte *db, TCHSTS *tsts, int xsp, int ysp, int xep, word mes,
                                    byte type, byte type1, byte F_Type)
```

[Arguments]

```
     byte      *db      :IN/OUT Date buffer     [8]
     TCHSTS   *tsts     :OUT    Touch status information
     int      xsp       :IN     Start point of display    Horizontal
     int      ysp       :IN     Start point of display    Vertical
     int      xep       :IN     End point of display    Horizontal
     word     mes       :IN     Guidance message number
                                  No message for 0xfffe
     byte     type      :IN     Day of the week display: preset/non
                                  WEEK_ON:    Present
                                  WEEK_OFF:   None
     byte     type1     :IN     Display format
                                  IN_MODE:    For input
                                  DISP_MODE:  For display
                                  DISP_YM:    Year-Month
                                  DISP_MY:    Month-Year
                                  IN_YM:      Year-Month input

     byte     F_Type    :IN     Font type
```

[Return values] If a touch area other than the keyboard area is touched, this function returns KEY_NONE.

[Description] Edits the date buffer using the Calendar keyboard. This function uses the Calendar keyboard to update the date buffer specified by “db”. The function also displays the date at specified coordinates at the same time.

- Touch functions -

[Function name] `LibInputTime`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    word LibInputTime(TCHSTS *tsts,LPTIMEKEYBCTRL lptbl,bool IsDispKeyb)
```

[Arguments]

```
    TCHSTS    *tsts              :IN/OUT Touch status information
    LPTIMEKEYBCTRL     lptbl     :IN/OUT Edit target time data buffer
    bool      IsDispKeyb         :IN  Draw/not draw a keyboard background.
                                      TRUE:   Draw.
                                      FALSE:  Not draw.
```

[Return values] `word      Status and key code`

```
              BIT15       Guidance display for "1".
              BIT14 to 12 Undefined
              BIT8 to 11 (4 bits)
                  CLKKEYB_RSLT_NONE:       NO EVENT
                  CLKKEYB_RSLT_NEXTITEM:  Performed a Minute setting.
                  CLKKEYB_RSLT_PREVITEM:  The left key was pressed when the
                                          cursor was at "0".
                  CLKKEYB_RSLT_OUTOFKEYB: The invalid area was touched.
                  CLKKEYB_RSLT_KEYBSWAP:  The time bar switch button was
                                          touched.
              BIT0 to 7   The virtual key code (CLKKEY_XXX) (See define.h)
```

[Description] Sets the time to the specified time array. Exceeding the range, moving to next item, or touching to the time bar exits the processing. The input value is five-byte character string in 'HHMMA' format (System time expression) ('A'(a) is entered to A for AM (morning), 'P'(p) is entered to A for PM. This "A" has no mean when it is 24 hours format.) When it is displayed, the appropriate format following to the system settings is used. The selection of AM/am is automatically performed depending on the state of the 5-th byte.

- User has to initialize (IsDispKeyb=TRUE, any others) xsp, ysp, xep, yep, font, csrpos when calling them first time. At this time, LibGetTime displays a keyboard at specified position, moves a cursor to the position specified by "csrpos", and enters the edit state. When IsDispKeyb=FALSE, the process is performed as the keyboard graphics exist.
- When entering this main process, lptbl->timbuf is not initialized. So, it is necessary to set the appropriate character string when initializing.
- Using LibChkTime, you can check if this character string is a correct time. The decision to close the keyboard has to be done by user application. So, arrange a judgment for closing the keyboard in the application according to the specification.
- Each time when calling this keyboard, the current cursor position is returned to "csrpos" member of this structure. When it is "-1", the keyboard will not be displayed.

- Event occurrence

  The occurrence condition of NEXTITEM(PREVITEM) by the result returned from LibInputTime is as follow.

  Right key or a numeric key is pressed when the cursor locates at right edge of the minute item.
  Left key is pressed when the cursor locates at left edge of the hour item.
  NEXT key is pressed when the cursor locates at the minute item. (NEXTITEM only)

  In the preset specification, when Right/Left key is pressed while the cursor locates at an edge of the minute item, it results NOP. However, be sure that it always issues the event mentioned above and steps out from the function in order to maintain the generality of the library. Therefore, the caller should refer the escaping key information (low-order 8-bit of the result) and manipulate whether making it NOP or not.

  The cursor can be controlled by existprevitem/existnextitem.

  If making them TRUE, the cursor is turned off when stepping out from this library with CLKKEYB_RSLT_NEXTITEM/PREVITEM.

* About cursor display
It is possible that the keyboard is closed without turning off a cursor. So, be sure that the cursor is always turned off when closing the keyboard (specification).

Internal variables

`csrpos:`	This is a member to indicate a cursor position. This cursor position does not correspond exactly to the actual cursor display, but it corresponds in the character string 'HHMMA'. (That is, ":" is not included in the calculation.) The last "A" is used only to display the input board AM/PM.

If the hardware icon "Menu Bar" is touched during this process, the menu bar process is performed internally and never goes back to higher order (not exit the process). Thus, it is not possible to assign the specific functions that correspond to the menu bar. (As of May 31, 1999)

[Examples of usage]

```
#define STIME_X1    78
#define STIME_Y1    19
#define STIME_X2    120
#define STIME_Y2    30
#define MES_OFFY    2

void main()
{
    TIMEKEYBCTRL    ptbl;
    TCHSTS          tsts;
    word  result;
    bool  Disp;
    byte  timebuf[8];

    LibClrDisp();
    LibPutDisp();

    LibTchStackClr();
    LibTchStackPush(NULL);
    LibTchStackPush(TchHardIcon);
    LibTchInit();

    memcpy(timebuf,"1813P",5);

    ptbl.xsp=STIME_X1+5;          /* Left top   Horizontal  */
    ptbl.ysp=STIME_Y1+MES_OFFY;    /* Left top  Vertical  */
    ptbl.xep=STIME_X2-1;           /* Right bottom  Horizontal  */
    ptbl.yep=STIME_Y2-1;           /* Right bottom  Vertical  */
    ptbl.timbuf=(byte far *)timebuf;    /* Time buffer to edit.  */
    ptbl.font=IB_PFONT1;           /* Proportional normal */
    ptbl.existnextitem=TRUE;   /* Next item=Present  */
    ptbl.existprevitem=FALSE;    /* Previous item=None  */
    ptbl.csrpos=0;      /* The initial cursor position is at the top. */
    ptbl.enabletimebar=FALSE;     /* Time bar button=None  */
    ptbl.guide=NULL;      /* Guidance strings=None */

    for(Disp=TRUE;;Disp=FALSE){ /*Drawing background at the first time.*/
        result = LibInputTime(&tsts,(LPTIMEKEYBCTRL)&ptbl,Disp);
        switch((result>>8) & 0x0F){

   case CLKKEYB_RSLT_NONE:          /* NO EVENT */
       break;
   case CLKKEYB_RSLT_NEXTITEM:      /* After the minute was set. */
      break;
   case CLKKEYB_RSLT_PREVITEM:      /* When the left key is pressed */
                                    /* while the cursor is at "0". */
       break;
   case CLKKEYB_RSLT_OUTOFKEYB:  /* Touches the invalid area. */
       break;
   case CLKKEYB_RSLT_KEYBSWAP:  /* Touches the time bar switch button. */
       break;
   default:
       break;

        }
    }
}
```

- Touch functions -

[Function name] `LibInputTimeBar`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibInputTimeBar(TCHSTS *tsts,LPTIMEKEYBCTRL lpstbl,LPTIMEKEYBCTRL lpetbl)
```

[Arguments]

```
TCHSTS *tsts                 :IN/OUT  Touch status information
LPTIMEKEYBCTRL        lpstbl :IN   Time array(start time)
LPTIMEKEYBCTRL        lpetbl :IN   Time array(end time)
```

[Return values]

```
word   Key code and time software keyboard control structure.
        Internal variable (This is the same as LibInputTime.)
```

[Description] Outputs a time bar keyboard only for the term input, and performs the term input. Basically, the input parameters are the same as LibInputTime. Though two structures are used since this is for the term input.

• Handling of cursor coordinate

The structure's value for a start point is only valid as a cursor coordinate. The position of the start point can be 0 to 3, and the end point can be 4 to 7. LibInputTimeBar refers lpstbl->csrpos and outputs a cursor to that position. The cursor position after setting can be write and back to both start point structure and end point structure. (However, only the cursor position for the start point side is referred when calling.) Therefore, when closing TimeBar once and outputting the time input keyboard again, the cursor position has to be corrected and called. (When exceeding the range, "0" position is output as default.)

• Time bar management member

LibInputTimeBar manages the time bar status separately from timbuf because of the specification of ZX-483. For this management, two members, barpos and barshift, are used. When the bar is not displayed on the screen, both lpstbl->barpo and lpetbl->barpos take a "-1", and the value is finalized by dragging the bar. (The start position is lpstbl->barpos, and the end position is lpetbl->barpos.) Therefore, first time when calling this function, set those two parameters to "-1" before calling the function. Then, the time bar input board is displayed without displaying the time bar. (However, even either is "-1", it is illegal. So the display will not be performed.) The barshift indicates the shift status of the bar; left, center or right. "1" is for Center, "0" is for Left (earlier), and "2" is for Right (later) respectively. Set this to "1" for the first opening. The barshift for the start point side is also referred.

• Event occurrence

The occurrence condition of NEXTITEM (PREVITEM) by the result returned from LibInputTimebar is as follow:

A numeric key is pressed when the cursor locates at an edge of the minute item.
NEXT key is pressed when the cursor locates at the minute item. (NEXTITEM only)

In the present specification, it specifies that the cursor does not move but the relevant column is rewritten when pressing a numeric key while the cursor locates at an edge of the minute item. However, be sure that it always issues the event mentioned above and steps out from the function in order to maintain the generality of the library. Because of this, the caller has to proceed whether returning the control again or not. It is possible to perform a branch judgement using the escaping key types stored in the low-order 8-bit of the result. At that time, the cursor can be controlled by existprevitem/existnextitem.

* Similar to the LibInputTime, it is possible that the keyboard is closed without turning off the cursor. So, be sure that the cursor is always turned off when closing the keyboard (specification).

• existnextitem member
This value is automatically set to lpstbl->existnextitem=FALSE/lpetbl->existprevitem=FALSE when this function is called. (When moving both items, the cursor is kept outputting.) The movement of lpstbl->lpetbl between the items is automatically performed in the library. When the cursor moves to lpstbl -> lpetbl, the NEXTITEM event is generated. The cursor position at that time is automatically moved to the tens digit of the hour item of lpetbl.

If the hardware icon "Menu Bar" is touched during this process, the menu bar process is performed internally and never goes back to higher order (not exit the process). Thus, it is not possible to assign the specific functions that correspond to the menu bar. (As of May 31, 1999)

- Touch functions -

[Function name] `LibInputTerm`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibInputTerm(TCHSTS *tsts,LPTIMEKEYBCTRL lpstbl,LPTIMEKEYBCTRL lpetbl)
```

[Arguments]

```
TCHSTS      *tsts              :IN/OUT Touch status information
LPTIMEKEYBCTRL  lpstbl         :IN     Time array(start time)
LPTIMEKEYBCTRL  lpetbl         :IN     Time array(end time)
```

[Return values]

```
word  Key code and time software keyboard control structure.
      Internal variable (This is the same as LibInputTime.)
```

[Description] Outputs the time bar keyboard only for term input, and performs the general processing of term input.

* Similar to the LibInputTime, it is possible that the keyboard is closed without turning off the cursor. So, be sure that the cursor is always turned off when closing the keyboard (specification).

- Touch functions -

[Function name] `LibSKeyRev`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibSKeyRev(TCHTBL far *sk_tbl,word obj_cd,byte sw)
```

[Arguments]

```
TCHTBL far  *sk_tbl   :IN  Information for touch table
word        obj_cd    :IN  Object code
byte        sw        :IN  Key action   ON: Down (Pressed) state
                                        OFF: Up state.
```

[Return values] `None`

[Description] Provides the pressed appearance to the software keyboard.
Checks if the object code indicated by "obj_cd" is included in the touch table "sk_tbl", and represents a matched coordinate position.

[Note] In order to provide fast processing, the boundary scan is not performed. Therefore, "obj_cd" has to be included in "sk_tbl[]".

- Touch functions -

[Function name] `LibSKeyIsCd`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    #include    "l_define.h"
    #include    "l_libc.h"
    bool LibSKeyIsCd(TCHTBL far *sk_tbl,word obj_cd)
```

[Arguments]

TCHTBL far	*sk_tbl	:IN	Information for touch table
word	obj_cd	:IN	Object code

[Return values]

```
Inspection result     TRUE:   Present
                      FALSE:   None
```

[Description] Checks whether the object code exists. Checks if the object code indicated by "obj_cd" is included in the touch table "sk_tbl".

- Touch functions -

[Function name] `LibIconMoveDown`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
void LibIconMoveDown(byte far *inbuf, byte *workbuf, byte kind)
```

[Arguments]
```
byte far  *inbuf              :IN     Graphic pattern for reverse.
byte      *workbuf            :IN     Work buffer
byte      kind                :IN     Icon type
```

[Return values] `None`

[Description] Provides the pressed appearance to the icon. (Sunken state)

- Touch functions -

[Function name] `LibIconMoveUp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibIconMoveUp(byte far *inbuf, byte *workbuf, byte kind)
```

[Arguments]

```
byte far  *inbuf            :IN   Graphic pattern for reverse.
byte      *workbuf          :IN   Work buffer
byte      kind              :IN   Icon type
```

[Return values] `None`

[Description] Provides the up-transition appearance to the icon. (An appearance for the icon released from the pressed state.)

- Touch functions -

[Function name] `LibBkSampleInit`

[Syntax]

```
#include "define.h"
#include   "libc.h"
void LibBkSampleInit(BK_SMPL_TCH far *t_tbl)
```

[Arguments]

```
BK_SMPL_TCH far *t_tbl   :IN  Touch coordinate table for ESC icon [2]
```

[Return values] `None`

[Description] Initializes the break key sample. (For the main system)
Assigns two touch coordinates for the ESC icon to "t_tbl[]". If there is only one coordinate, assign the same value to two coordinates.

[Note] A value that crosses the "0" dot cannot be assigned to t_tbl[]. If it does not cross, even a negative figure can be used. This function is used to perform FLASH accessing process in the system. (Search function, etc.) Executes LibBkSampleInitSub() with a cause specification for the initialization process during communication process because the break cause to be valid is different.

- Touch functions -

[Function name] `LibBkSampleCheck`

[Syntax]

```
#include "define.h"
#include "libc.h"
byte LibBkSampleCheck(bool *passed)
```

[Arguments]

```
bool     *passed            :OUT  1 sec passed: yes/no
                                TRUE: 1 sec passed
                                FALSE: Not passed yet
```

[Return values]

```
byte    Inspection result          ON: Break cause occur
                                  OFF: No break cause yet
```

[Description] Monitors the break key sample status. Monitors if it has passed one second since the initialization by LibBkSampleInit() and LibBkSampleInitSub() was performed, and if the break cause pre-set has been occurred.

[Examples of usage]

```
BK_SMPL_TCH BkTchTbl[2];
byte        bk_chk;
bool        passed,mes_done;

/* Coordinates of ESC icon1 for break (Assigns the ESC hardware -icon)*/
BkTchTbl[0].x1 = TchHardIcon[7].x1;
BkTchTbl[0].y1 = TchHardIcon[7].y1;
BkTchTbl[0].x2 = TchHardIcon[7].x2;
BkTchTbl[0].y2 = TchHardIcon[7].y2;

/* Coordinates of ESC icon2 for break (Assigns the ESC hardware -icon) */
BkTchTbl[1].x1 = TchHardIcon[7].x1;
BkTchTbl[1].y1 = TchHardIcon[7].y1;
BkTchTbl[1].x2 = TchHardIcon[7].x2;
BkTchTbl[1].y2 = TchHardIcon[7].y2;


passed   = FALSE;   /* Elapsed 1 sec = Yet*/
mes_done = FALSE;   /* Message displayed = Yet */

LibBkSampleInit(BkTchTbl);  /* Initializing a break-key sample. */

while(1){

    bk_chk = LibBkSampleCheck(&passed); /* Checking break-key. */
```

```
  if(mes_done == FALSE){
       if(passed == TRUE){ /* 1 sec passed!! */
/* Keep displaying the "processing" message if 1 second is elapsed. */
LibWinIcnMsg(ICON_COFFEE,336,0);    /* Please_wait...    */
LiPutDisp();         /* Makes the display valid. */
LibCloseWindow();            /* LibWinIcnMsg(0) minute(s) */

mes_done = TRUE;
       }
  }

  if(bk_chk == ON){   /* Break (abort process) occur!*/
       LibWinIcnMsg(ICON_BIKKURI,387,2);    /* Stopped! */
           break;  /* Escapes from the process. */
     }


          /****************************************/
          /*   Flash access processes: Search, etc.  */
          /*                  ●                  */
          /*                  ●                     */
          /*                  ●                  */
          /*                  ●                  */
          /****************************************/

}
```

- Touch functions -

[Function name] `LibBkSampleInitSub`

[Syntax]

```
     #include  "define.h"
     #include  "libc.h"
     #include  "l_define.h"
     #include  "l_libc.h"
     void LibBkSampleInitSub(BK_SMPL_TCH far *t_tbl,byte b_smp)
```

[Arguments]

```
     BK_SMPL_TCH far *t_tbl       :IN   Touch oordinate table for ESC [2]
     byte      b_smp              :IN   Effective break cause (assigns with OR).
                                       IX_BLD1MSG: BLD is vald.
                                       IX_CRADLE: Cradle key is valid.
                                       IX_ESCBRK: ESC touch is valid.
```

[Return values] `None`

[Description] Initializes the break key sample. (Body process.)
Assigns two touch coordinates for the ESC icon to "t_tbl[]". If there is only one coordinate, assign the same value to two coordinates.

[Note] A value that crosses the "0" dot cannot be assigned to t_tbl[]. If it does not cross, even a negative figuare can be used.

- Touch functions -

[Function name] `LibBlockIconClick`

[Syntax]

```
#include "define.h"
#include "libc.h"
bool LibBlockIconClick(const T_ICON far *icon,TCHSTS *tsts,byte opt)
```

[Arguments]

```
const T_ICON far   *icon      :IN     Icon definition information
TCHSTS    *tsts               :IN     Touch status information
byte      opt                 :IN     Option(s)
                                  B_ICON_LEFT: Left-aligned type
                                  B_ICON_CENTER: Centered type
                                  B_ICON_RIGHT: Right-aligned type
```

[Return values]

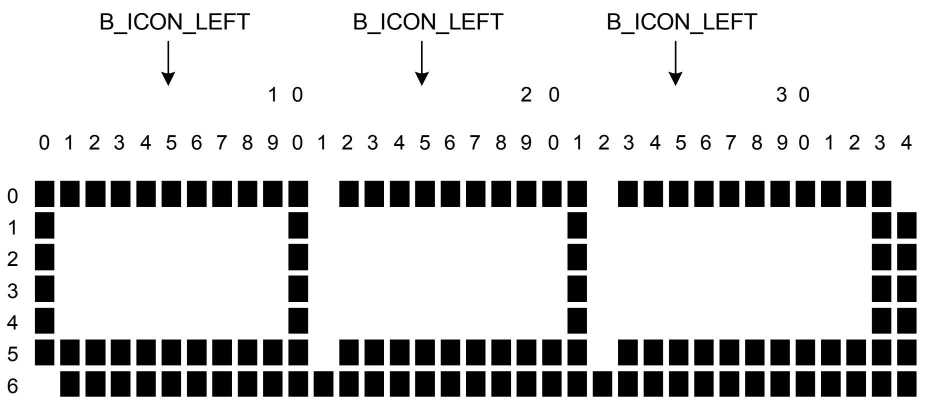
```
bool   Execution result   TRUE: Finalized the pressed state.
                                           (Break occurs.)
                          HALF: Repeating.
                          FALSE: Not finalized.
```

[Description] Controls the click operations of the block-type icon (link type) such as a header toolbar. This function is applicable to repeat operations depending on the action designation.

[Note] 1. Adds ACT_REPEAT to the action designation in order to make the repeat operation valid. (Example: ACT_ICON | ACT_REPEAT)
2. The size of an icon is up to 256 bytes.
3. This function uses following global variables. Because of this, the use of those variables by functions in other bank is prohibited until the variables are saved to the common external variable file "libdata.c".

```
byte    _IconBuf_[B_ICON_BUF];
byte    _LstRpt_;
```

[Supplement] The following shows how to specify coordinates of a touch area used in LibBlockIconClick.



```
B_ICON_LEFT:      0, 0 - 10, 6
B_ICON_CENTER:    11, 0 - 21, 6
B_ICON_RIGHT:     22, 0 - 34, 6
```

[Examples of usage]

```
#define OBJ_HEAD00      0xed00
#define OBJ_HEAD01      0xed01
#define OBJ_HEAD02      0xed02
#define OBJ_HEAD03      0xed03
#define OBJ_HEAD04      0xed04

#define X   14
#define Y   0

static TCHTBL far TchHead[] =   /* Header icon touch information */
{
   X+  0, Y+ 0, X+ 28, Y+ 11, ACT_ICON, OBJ_HEAD00,0x0000, /* List */
   X+ 29, Y+ 0, X+ 41, Y+ 11, ACT_ICON | ACT_REPEAT, OBJ_HEAD01, 0x0000,
                                                         /* Previous page*/
   X+ 42, Y+ 0, X+ 54, Y+ 11, ACT_ICON | ACT_REPEAT, OBJ_HEAD02,0x0000,
                                                             /* Next page*/
   X+ 55, Y+ 0, X+ 83, Y+ 11, ACT_ICON,  OBJ_HEAD03,0x0000, /* Edit */
   X+ 84, Y+ 0, X+113, Y+ 11, ACT_ICON,  OBJ_HEAD04,0x0000, /* New  */
    0,0,0,0,ACT_NONE,OBJ_END,0x0000
};

static T_ICON far Ticon00 = {&TchHead[0],NULL,NULL,0x00}; /* List */
static T_ICON far Ticon01 = {&TchHead[1],NULL,NULL,0x00};
                                                        /* Previous page*/
static T_ICON far Ticon02 = {&TchHead[2],NULL,NULL,0x00}; /* Next page*/
static T_ICON far Ticon03 = {&TchHead[3],NULL,NULL,0x00}; /* Edit */
static T_ICON far Ticon04 = {&TchHead[4],NULL,NULL,0x00}; /* New  */

void main()
{
    TCHSTS      tsts;

    LibTchStackClr();
    LibTchStackPush(NULL);
    LibTchStackPush(TchHardIcon);
    LibTchStackPush(TchHead);
    LibTchInit();

    LibClrDisp();
    LibPutFarData(X,Y,145);     /* 114 * 12 */
    LibPutMessageCenter( 49, X+  1, X+ 27,  Y+  2,IB_PFONT1);  /* List */
```

```
     LibPutMessageCenter( 15, X+ 56, X+ 82,  Y+  2,IB_PFONT1);  /* Edit */
     LibPutMessageCenter( 14, X+ 85, X+111,  Y+  2,IB_PFONT1);  /* New  */
     LibPutDisp();

     for(;;){
         LibTchWait(&tsts);

         switch(tsts.obj){

    case OBJ_HEAD00:
        if(LibBlockIconClick(&Ticon00,&tsts,B_ICON_LEFT)==TRUE){
            LibPutMsgDlg2("List!!");
        }
        break;

    case OBJ_HEAD01:
        if(LibBlockIconClick(&Ticon01,&tsts,B_ICON_CENTER)!=FALSE){
            LibPutMsgDlg2("Before!!");
        }
        break;

    case OBJ_HEAD02:
        if(LibBlockIconClick(&Ticon02,&tsts,B_ICON_CENTER)!=FALSE){
            LibPutMsgDlg2("Next!!");
        }
        break;

    case OBJ_HEAD03:
        if(LibBlockIconClick(&Ticon03,&tsts,B_ICON_CENTER)==TRUE){
            LibPutMsgDlg2("Edit!!");
        }
        break;

    case OBJ_HEAD04:
        if(LibBlockIconClick(&Ticon04,&tsts,B_ICON_RIGHT)==TRUE){
            LibPutMsgDlg2("New!!");
        }
        break;

    default:
        break;
         }
     }

}
```

- Touch functions -

[Function name] `LibRepOff`

[Syntax]

```
     #include  "define.h"
     #include  "libc.h"
     void LibRepOff(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Turns off the touch repeat function.

- FLASH functions -

[Function name] `LibFileFindNext`
[Function name] `LibLFileFindNext`

[Syntax]
```
#include  "define.h"
#include  "libc.h"
bool LibFileFindNext(const FILE_BUF *fb, FILE_INFO *finf, byte search);
bool LibLFileFindNext(const LFILE_BUF far *fb, FILE_INFO *finf, byte search);
```

[Arguments]
```
FILE_BUF    *fb             :IN   File buffer(LibFileFindNext)
LFILE_BUF far *fb           :IN   File buffer(LibLFileFindNext)
FILE_INFO   *finf           :IN/OUT File information
byte       search           :IN   Search conditions
```

[Return values]
```
bool   Result   TRUE: Has data.
                FALSE:  No data.
```

[Description] Searches for next data.
Performs the data search from the current data pointer on FLASH specified by "finf->fp" to the next direction based on the search conditions. If next data is found, the function updates “finf->fp”, and returns TRUE.

When 0xffff is set to “finf->fp”, searches for data from the beginning.
When 0xfffe is set to “finf->fp”, searches for the last data.

[Examples of usage]
```
FILE_BUF    fb;
FILE_INFO   finf;
bool        f_handle;

fb.main_entry= 0x??;    /* Mode         */
fb.sub_entry = 0x??;    /* Sub-mode      */
fb.scrt_info = 0x80;    /* Open area      */

finf.fp      = 0xffff;          /* From top      */
finf.kind    = 01;     /* Binary designation   */

f_handle = LibFileFindNext(&fb,&finf,0x00);

if(f_handle==TRUE){
    LibFileRead(&fb,&finf);     /* Data read */
}
```

- FLASH functions -

[Function name] `LibFileFindPrev`
[Function name] `LibLFileFindPrev`

[Syntax]
```
     #include  "define.h"
     #include  "libc.h"
     bool LibFileFindPrev(const FILE_BUF *fb, FILE_INFO *finf, byte search)
     bool LibLFileFindPrev(const LFILE_BUF far *fb, FILE_INFO *finf, byte search)
```

[Arguments]
```
     FILE_BUF      *fb            :IN  File buffer(LibFileFindPrev)
     LFILE_BUF far *fb            :IN  File buffer(LibLFileFindPrev)
     FILE_INFO     *finf          :IN/OUT   File information
     byte          search         :IN  Search conditions
```

[Return values]
```
                   bool   Result   TRUE: Has data.
                                   FALSE:  No data.
```

[Description] Searches for previous data.
Performs the data search from the current data pointer on FLASH specified by "finf->fp" to the previous direction based on the search conditions. If previous data is found, the function updates “finf->fp”, and then returns TRUE.

[Examples of usage]
```
     FILE_BUF    fb;
     FILE_INFO   finf;
     bool        f_handle;

     fb.main_entry= 0x??;   /* Mode          */
     fb.sub_entry = 0x??;   /* Sub-mode       */
     fb.scrt_info = 0x80;   /* Open area       */

     finf.kind    = FILE_KIND_BIN;   /* Binary specification  */

     f_handle = LibFileFindPrev(&fb,&finf,0x00);

     if(f_handle==TRUE){
         LibFileRead(&fb,&finf);     /* Data read */
     }
```

- FLASH functions -

[Function name] `LibFileFindNextExt`
[Function name] `LibLFileFindNextExt`

[Syntax]

```
     #include  "define.h"
     #include  "libc.h"
     bool LibFileFindNextExt(const FILE_BUF *fb, FILE_INFO *finf, byte search)
     bool  LibLFileFindNextExt(const  LFILE_BUF  far  *fb,  FILE_INFO  *finf,  byte
search)
```

[Arguments]

```
     FILE_BUF      *fb                 :IN/OUT File buffer(LibFileFindNextExt)
     LFILE_BUF far *fb                 :IN/OUT File buffer(LibLFileFindNextExt)
     FILE_INFO     *finf               :IN/OUT File information
     byte          search              :IN     Search conditions
```

[Return values]
```
bool   Result   TRUE: Has data.
                HALF: Has data but not perfect match.
                FALSE:No data.
```

[Description] Searches for next data. (For extension.)
Performs the data search from the current data pointer on FLASH specified by "finf->fp" to the next direction based on the search conditions. If next data is found, the function updates “finf->fp”, and then returns TRUE.

When 0xffff is set to “finf->fp”, searches for data from the beginning.
When 0xfffe is set to “finf->fp”, searches for the last data.

Differently from LibFileFindNext(), this function also supports data other than the perfect matching data.

- FLASH functions -

[Function name] `LibNextSearchCld`
[Function name] `LibLNextSearchCld`

[Syntax]

```
#include "define.h"
#include "libc.h"
bool LibNextSearchCld(FILE_BUF *fb, FILE_INFO *finf, byte search)
bool LibLNextSearchCld(LFILE_BUF far *fb, FILE_INFO *finf, byte search)
```

[Arguments]

```
FILE_BUF  *fb                :IN/OUT File buffer(LibNextSearchCld)
LFILE_BUF far *fb            :IN/OUT File buffer(LibLNextSearchCld)
FILE_INFO   *finf            :IN/OUT File information
byte      search             :IN     Search conditions
```

[Return values]
```
bool    Result   TRUE: Has data.
                 HALF: Has data but not perfect match.
                 FALSE: No data.
```

[Description] Searches for next data. (For Calendar.)
This function has the same function with LibFileFindNextExt(). However, when FALSE is returned, finf->fp is destroyed.

- FLASH functions -

[Function name] `LibFileRead`
[Function name] `LibLFileRead`

[Syntax]

```
#include  "define.h"
#include  "libc.h"
bool LibFileRead(FILE_BUF *fb, const FILE_INFO *finf)
bool LibLFileRead(LFILE_BUF far *fb, const FILE_INFO *finf)
```

[Arguments]

```
FILE_BUF *fb                :IN/OUT File buffer(LibFileRead)
LFILE_BUF far *fb           :IN/OUT File buffer(LibLFileRead)
const FILE_INFO    *finf    :IN     File information
```

[Return values]

```
bool  Result    TRUE: Normal
                FALSE: Error
```

[Description] Reads data from the FLASH memory.
This function reads FLASH memory data from the data pointer specified by “finf->fp” and sets it in the buffer specified by “fb”. Position in the buffer may vary depending on the mode/sub-mode.

[Examples of usage]

```
FILE_BUF    fb;
FILE_INFO   finf;
bool        f_handle;

fb.main_entry= 0x??;    /* Mode          */
fb.sub_entry = 0x??;    /* Sub-mode       */
fb.scrt_info = 0x80;    /* Open area       */

finf.fp      = 0xffff;          /* From top      */
finf.kind    = FILE_KIND_BIN;   /* Binary specification   */

f_handle = LibFileFindNext(&fb,&finf,0x00);

if(f_handle==TRUE){
    LibFileRead(&fb,&finf);     /* Data read */
}
```

- FLASH functions -

[Function name] `LibFileWrite`
[Function name] `LibLFileWrite`

[Syntax]
```
#include "define.h"
#include "libc.h"
bool LibFileWrite(const FILE_BUF *fb, FILE_INFO *finf)
bool LibLFileWrite(const LFILE_BUF far *fb, FILE_INFO *finf)
```

[Arguments]
```
const FILE_BUF      *fb        :IN     File buffer(LibFileWrite)
const LFILE_BUF far *fb        :IN     File buffer(LibLFileWrite)
FILE_INFO    *finf             :IN/OUT File information
```

[Return values]
```
bool   Result    TRUE: Normal
                 FALSE: Error
```

[Description] Writes data to the FLASH memory.
This function writes the contents in the buffer specified by "fb" to the data pointer specified by "finf->fp", and then sets a new data pointer to "finf->fp".
If 0xffff is set to "finf->fp", data is registered newly.
If the function fails to write data, it returns FALSE.

[Note] When writing data to the FLASH memory, the data pointer changes even if it is just a correction of existing data.
Therefore, if the data pointer is held in buffer such as list buffer, the changed finf->fp after calling this function must be reflected.

[Examples of usage]
```
bool    f_handle;

finf.fp     = 0xffff;
finf.kind   = FILE_KIND_TEXT;

f_handle = LibFileWrite(&fb, &finf);
if(f_handle == FALSE){
    /*Error process */
}
```

- FLASH functions -

[Function name] `LibFileCorect`
[Function name] `LibLFileCorect`

[Syntax]

```
#include "define.h"
#include "libc.h"
bool LibFileCorect(const FILE_BUF *fb, FILE_INFO *finf, byte type)
bool LibLFileCorect(const LFILE_BUF far *fb, FILE_INFO *finf, byte type)
```

[Arguments]

```
const FILE_BUF   *fb          :IN  File buffer(LibFileCorect)
const LFILE_BUF  far *fb      :IN  File buffer(LibLFileCorect)
FILE_INFO   *finf             :IN  File information
byte        type      :IN  Type 0: Normal correction
                                 1: Corrects without changing the pointer.
```

[Return values]

```
bool    Result      TRUE: Normal
                    FALSE: Error
```

[Description] Writes the change of the data pointer to the FLASH memory with the option specification.
This function writes the contents in the buffer “fb” to the data pointer specified by “finf->fp”.
When "type" is set to "1", the data pointer does not change.
If the function fails to write data, it returns FALSE.

[Note] Supports only for correction.
The normal operation of this function can be performed only for some limited modes such as SCHEDULE mode.
See the BIOS document when using a mode with this function.
Normally, LibFileWrite() should be used.

[Examples of usage]

```
bool    f_handle;

f_handle = LibFileChangeTodo(&fb, &finf);
if(f_handle == FALSE){
    /*Error process */
}
```

- FLASH functions -

[Function name] `LibFileRemove`
[Function name] `LibLFileRemove`

[Syntax]

```
#include "define.h"
#include "libc.h"
bool LibFileRemove(const FILE_BUF *fb, const FILE_INFO *finf)
bool LibLFileRemove(const LFILE_BUF far *fb, const FILE_INFO *finf)
```

[Arguments]

```
const FILE_BUF     *fb       :IN     File buffer(LibFileRemove)
const LFILE_BUF far *fb      :IN     File buffer(LibLFileRemove)
const FILE_INFO    *finf     :IN     File information
```

[Return values]

```
bool   Result  TRUE: Normal
               FALSE: Error
```

[Description] Deletes data (1 record) in the FLASH memory.
Deletes data of the data pointer specified by “finf->fp”.

[Note] "fb" in the first argument is not used.

[Examples of usage]

```
finf.fp = lst_buf[3].fp;
LibFileRemove(&fb, &finf);
```

- FLASH functions -

[Function name] `LibFileRemoveAll`
[Function name] `LibLFileRemoveAll`

[Syntax]
```
#include "define.h"
#include "libc.h"
bool LibFileRemoveAll(const FILE_BUF *fb)
bool LibLFileRemoveAll(const LFILE_BUF far *fb)
```

[Arguments]
```
const FILE_BUF     *fb      :IN     File buffer(LibFileRemove)
const LFILE_BUF    far *fb  :IN     File buffer(LibLFileRemove)
```

[Return values]
```
bool   Result    TRUE: Normal
                 FALSE: Error
```

[Description] Deletes all data (multiple data) in the FLASH memory.
Deletes all data specified by “fb->main_entry, and fb->sub_entry” in units of modes/sub-modes.

[Note] Be sure to set the following information.
```
fb->fsb_main_entry_: Mode number
fb->fsb_sub_entry_: Sub-mode number (All data when 0x00)
fb->fsb_scrt_info_: Secret zone
```

This will not delete both Secret and Open modes. Only an area specified by "fb->scrt_info" is a target.

[Examples of usage]
```
FILE_BUF    fb;

fb.main_entry   = 0x??;     /* Mode         */
fb.sub_entry    = 0x??;     /* Sub-mode      */
fb.scrt_info    = 0x80;     /* Open area      */

LibFileRemoveAll(&fb);      /* Delete all data   */
```

- FLASH functions -

[Function name] `LibGetFileInfo`
[Function name] `LibLGetFileInfo`

[Syntax]

```
    #include  "define.h"
    #include  "libc.h"
    void LibGetFileInfo(FILE_BUF *fd, FILE_INFO *fi)
    void LibLGetFileInfo(LFILE_BUF far *fd, FILE_INFO *fi)
```

[Arguments]

FILE_BUF	*fb	:OUT	File buffer(LibGetFileInfo)
LFILE_BUF	far *fb	:OUT	File buffer(LibLGetFileInfo)
FILE_INFO	*finf	:IN	File information

[Return values] None

[Description] Gets the file information.
Reads the header information from the data pointer specified by “fi->fp”, and then outputs it to the following members of “fd”.

```
  fb->fsb_main_entry_:  Mode number
  fb->fsb_sub_entry_:   Sub-mode number
  fb->fsb_scrt_info_:   Secret zone
  fb->fsb_ararm_chk_:   Alarm check information
  fb->fsb_todo_chek_:   TODO check information
  fb->fsb_todo_rank_:   TODO rank information
```

- FLASH functions -

[Function name] `LibGetFileCnt`
[Function name] `LibLGetFileCnt`

[Syntax]
```
     #include   "define.h"
     #include   "libc.h"
     word LibGetFileCnt(FILE_BUF *fd)
     word LibLGetFileCnt(LFILE_BUF far *fd)
```

[Arguments]
```
     FILE_BUF    *fb             :IN     File buffer(LibGetFileCnt)
     LFILE_BUF   far *fb         :IN     File buffer(LibLGetFileCnt)
```

[Return values] `word` `Number of records`

[Description] Gets the number of records registered in the FLASH memory. The number of records to obtain is in the following units:
```
  fb->fsb_main_entry_: Mode number
  fb->fsb_sub_entry_:  Sub-mode number
  fb->fsb_scrt_info_:  Secret zone
  fb->fsb_todo_chek_:  TODO check information   *sub-mode ToDo only
  fb->fsb_todo_rank_:  TODO rank information    *sub-mode ToDo only
```

- FLASH functions -

[Function name] `LibGetFlash`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibGetFlash(void)
```

[Arguments] `None`

[Return values] `Total capacity`

[Description] Gets the total capacity of the FLASH memory.

- FLASH functions -

[Function name] `LibGetFreeBlock`

[Syntax]

```
     #include      "define.h"
     #include      "libc.h"
     word LibGetFreeBlock(void)
```

[Arguments] `None`

[Return values] `Free blocks`

[Description] Gets the number of free blocks of the FLASH memory.

- FLASH functions -

[Function name] `LibGetDataCond`

[Syntax]

```
#include  "define.h"
#include  "libc.h"
bool LibGetDataCond(void)
```

[Arguments] `None`

[Return values]

```
bool  Inspection result  TRUE: Normal
                         FALSE: Abnormal
```

[Description] Checks the FLASH data status.

- FLASH functions -

[Function name] `LibFileRemake`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
bool LibFileRemake(void)
```

[Arguments] `None`

[Return values]

`bool`	`Execution result`	`TRUE: Succeeded`
		`FALSE: Failed`

[Description] Executes the FLASH memory remaking process.

[Note] The BIOS that is called by this function checks the break key sample during processing. Be sure to initialize the break key before executing this function.

- FLASH functions -

[Function name] `LibFileExch`
[Function name] `LibLFileExch`

[Syntax]

```
#include  "define.h"
#include  "libc.h"
bool LibFileExch(const FILE_BUF *fb, FILE_INFO *finf, const word mvp)
bool LibLFileExch(const LFILE_BUF far *fb, FILE_INFO *finf, const word mvp)
```

[Arguments]

```
const FILE_BUF    *fb        :IN      File buffer(LibFileExch)
const LFILE_BUF   far *fb    :IN      File buffer(LibLFileExch)
FILE_INFO         *finf      :IN/OUT   File information
const word         mvp       :IN      Destination data pointer
```

[Return values]

```
bool    Execution result     TRUE: Succeeded
                             FALSE: Failed
```

[Description] Moves the pointer data specified by "finf->fp" to the "mvp" pointer. The data following to "mvp" are shifted one toward next direction.

- FLASH functions -

[Function name] `LibTelPtCnvrt`

[Syntax]

```
     #include  "define.h"
     #include  "libc.h"
     word LibTelPtCnvrt(word fp)
```

[Arguments]

```
     word     fp       :IN     Company data pointer
```

[Return values] `word  Personal data pointer`

[Description] Converts the Company data pointer in the Contacts mode specified by "fp" to the Personal data pointer, and returns the data.

- FLASH functions -

[Function name] `LibFileWriteCheckInit`

[Syntax]

```
     #include  "define.h"
     #include  "libc.h"
     void LibFileWriteCheckInit(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Performs the initial settings of LibFileWriteCheck().
Calls this function one time before executing LibFileWriteCheck().
LibFileWriteCheck() checks a capacity of the FLASH memory based on the FLASH capacity at the last execution of this function.

[Note] The normal operation of this function can be performed only for some limited modes such as EXPENSE mode.
See the BIOS document when using a mode with this function.

- FLASH functions -

[Function name] `LibFileWriteCheck`
[Function name] `LibLFileWriteCheck`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    bool LibFileWriteCheck(const FILE_BUF *fb)
    bool LibLFileWriteCheck(const LFILE_BUF far *fb)
```

[Arguments]

```
    const FILE_BUF     *fb       :IN     File buffer(LibFileWriteCheck)
    const LFILE_BUF    far *fb   :IN     File buffer(LibLFileWriteCheck)
```

[Return values]	bool	Result	TRUE: Write enabled.
			FALSE: Write disabled.

[Description] Checks whether data can be written to the FLASH memory.
Normally it is required to execute LibFileWrite() for checking the FLASH memory full. However, it is possible to check it using this function in advance.
Examines whether the buffer data specified by "*fb" can be written using LibFileWrite().

[Note] Be sure to execute LibFileWriteCheckInit() immediately before using this function.
The normal operation of this function can be performed only for some limited modes such as EXPENSE mode.
See the BIOS document when using the mode with this function.

- FLASH functions -

[Function name] `LibFileReadEx`
[Function name] `LibLFileReadEx`

[Syntax]
```
    #include    "define.h"
    #include    "libc.h"
    bool LibFileReadEx(FILE_BUF *fb, const FILE_INFO *finf,int maxblock)
    bool LibLFileReadEx(LFILE_BUF far *fb, const FILE_INFO *finf,int maxblock)
```

[Arguments]
```
    const FILE_BUF     *fb       :IN/OUT   File buffer(LibFileReadEx)
    const LFILE_BUF    far *fb   :IN/OUT   File buffer(LibLFileReadEx)
    FILE_INFO          *finf     :IN       File information
    Int                maxblock  :IN       The number of the blocks to read
                                           (maxblock >= 1)
```

[Return values]
```
    bool     Result     TRUE: Succeeded
                        FALSE: Failed
```

[Description] It reads specification block number data from FLASH.

[Note] The size of 1block is defined in "FILE_BLOCK_SIZE".

- Alarm functions -

[Function name] `LibAlarm`

[Syntax]

```
#include     "define.h"
#include     "libc.h"
void LibAlarm(void)
```

[Arguments] `None`

[Return values] `None`

[Description] In the past, this performed alarm coincidence processing, but the processing is now NOP as it is transferred to the alarm control module.

- Alarm functions -

[Function name] `LibNextAlmSet`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibNextAlmSet(void)
```

[Arguments] `None`

[Return values] `None`

[Description] In the past, this performed NEXT alarm setting, but the processing is now NOP as it is transferred to the alarm control module.

- Alarm functions -

[Function name] `LibInitAlarmFlg`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibInitAlarmFlg(void)
```

[Arguments] `None`

[Return values] `None`

[Description] In the past, this cleared alarm coincidence flags, but the processing is now NOP as it is transferred to the alarm control module.

- Alarm functions -

[Function name] `LibInitAlarmFlgCheck`

[Syntax]

```
#include "define.h"
#include "libc.h"
bool LibInitAlarmFlgCheck(void)
```

[Arguments] `None`

[Return values]	Inspection result	TRUE: Matched.
		FALSE: Unmatched.

[Description] In the past, this checked existence of alarm coincidence, but the processing is now NOP as it is transferred to the alarm control module.

- Alarm functions -

[Function name] `LibNextAlarmSet`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
bool LibNextAlarmSet(ALMAPL *almap, char type)
```

[Arguments]

```
ALMAPL    *almap    :IN/OUT  Alarm information table.
char      type      :IN    Setting types
```

[Return values]

```
Execution result
 TRUE: Normal end
 FALSE: Abnormal end (The value set exceeds the input range.)
```

[Description] In the past, this set the alarm that would sound next, but the processing is now NOP as it is transferred to the alarm control module.

- Alarm functions -

[Function name] `LibSetDailyAlarm`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
int LibSetDailyAlarm(char *tbp)
```

[Arguments]

```
char      *tbp      :IN     Time buffer
```

[Return values]

```
  Execution result
TRUE: Normal end
FALSE: Abnormal end (The value set exceeds the input range.)
```

[Description] Sets the daily alarm time.

- Alarm functions -

[Function name] `LibInitAlarm`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibInitAlarm(void)
```

[Arguments] `None`

[Return values] `None`

[Description] In the past, this cleared alarm settings, but the processing is now NOP as it is transferred to the alarm control module.

- Alarm functions -

[Function name] `LibGetAlarmInfo`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
bool LibGetAlarmInfo(FILE_BUF *afd, FILE_INFO *afi)
```

[Arguments]

```
FILE_BUF   *afd    :IN/OUT
FILE_INFO  *afi    :IN/OUT
```

[Return values]

```
Match data result      TRUE: Matched.
                       FLASE: Unmatched.
```

[Description] In the past, this acquired alarm information already set, but the processing is now NOP as it is transferred to the alarm control module.

- Alarm functions -

[Function name] `LibGetAlarmFlg`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
int LibGetAlarmFlg(void)
```

[Arguments] `None`

[Return values]

```
Inspection result   TRUE: ON
                    FALSE: OFF
```

[Description] In the past, this checked the on/off state of alarm switch, but the processing is now NOP as it is transferred to the alarm control module.

- Alarm functions -

[Function name] `LibGetDailyAlarm`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    #include    "l_define.h"
    #include    "l_libc.h"
    void LibGetDailyAlarm(char *tbp)
```

[Arguments]

```
    char      *tbp       :OUT    Time buffer
```

[Return values] `None`

[Description] Gets a daily alarm time.

- Alarm functions -

[Function name] `LibGetNextAlm`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibGetNextAlm(ALMAPL *almap, byte ifdel)
```

[Arguments]

```
ALMAPL    *almap    :IN/OUT Alarm information table.
byte       ifdel    :IN     Options
                              0:  Retain current alarm.
                              1:  Delete current alarm.
```

[Return values] None

[Description] In the past, this acquired the pointer to next alarm, but the processing is now NOP as it is transferred to the alarm control module.

- Alarm functions -

[Function name] `LibAlarmBuzzSet`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
byte LibAlarmBuzzSet(byte b_type)
```

[Arguments]

```
byte     b_type    :IN     Buzzer type
                         IB_ALMON_DILY:  For Daily
                         IB_ALMON_DATA:  For Schedule data
                         IB_ALMOFF: Releases buzzer setting
                         IB_ALMBZZ_NOW:  Gets buzzer status
                         IB_ALMPUSHWORK: Saves event management work.
                         IB_ALMPOPWORK: Restores event management work.
```

[Return values]

```
byte    Buzzer status   IB_ALMON_DILY: Daily alarm is sounding.
 (Only IB_ALMBZZ_NOW)   IB_ALMON_DATA: Schedule alarm is sounding.
                        IB_ALMOFF: Buzzer off.
```

[Description] Performs the buzzer-related controls during alarm matches.
At this point, if ON is set by IB_ALMON_DILY and IB_ALMON_DATA, the BLD message is suppressed until the setting is released by IB_ALMOFF.

[Note] Never use this function except for the alarm process.

- Alarm functions -

[Function name] `LibGetAlarmObj`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
#include   "l_define.h"
#include   "l_libc.h"
void LibGetAlarmObj(TCHTBL *t_tbl)
```

[Arguments]

```
TCHTBL      *t_tbl      :OUT    Touch table information
```

[Return values] `None`

[Description] In the past, this acquired alarm coincidence touch table information, but the processing is now NOP as it is transferred to the alarm control module.

- Alarm functions -

[Function name] `LibChkSysAlarm`

[Syntax]

```
     #include  "define.h"
     #include  "libc.h"
     void LibChkSysAlarm(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Corrects the system alarm data.
Checks the system alarm and corrects if the value is not normal.

- Date/Time functions -

[Function name] `LibGetDateTimeM`

[Syntax]

```
#include  "define.h"
#include  "libc.h"
byte LibGetDateTimeM(byte *d_data)
```

[Arguments]

```
byte    *d_data    :OUT    Year, month, day, hour, minute, second  [7]
```

[Return values] `The day of the week`

[Description] Gets the current date/time in the BCD format. Summer time (Daylight saving time) correction is provided.

[Examples of usage]

```
byte    tmp_bcd[7];

LibGetDateTimeM(tmp_bcd);
```

- Date/Time functions -

[Function name] `LibGetDateTime`

[Syntax]

```
#include "define.h"
#include "libc.h"
byte LibGetDateTime(byte *yearh, byte *yearl, byte *month, byte *day,
                       byte *hour, byte *minute, byte *second)
```

[Arguments]

byte	*yearh	:OUT	Year high
byte	*yearl	:OUT	Year low
byte	*month	:OUT	Month
byte	*day	:OUT	Day
byte	*hour	:OUT	Hour
byte	*minute	:OUT	Minute
byte	*second	:OUT	Second

[Return values] `The day of the week`

[Description] Gets the current date/time in the BCD format. Summer time (Daylight saving time) correction is provided.

[Examples of usage]

```
byte    tmp_bcd[7];
byte    dow;

dow = LibGetDateTime(
    &d_data[0],     /*  Year (High)  */
    &d_data[1],     /*  Year (Low)  */
    &d_data[2],     /*  Month  */
    &d_data[3],     /*  Day  */
    &d_data[4],     /*  Hour */
    &d_data[5],     /*  Minute */
    &d_data[6]      /*  Second  */
    );
```

- Date/Time functions -

[Function name] `LibGetDateTime2`

[Syntax]

```
#include  "define.h"
#include  "libc.h"
byte LibGetDateTime2(word *year2, byte *month2, byte *day2,
                       byte *hour2, byte *minute2, byte *second2)
```

[Arguments]

```
word    *year2      :OUT     Year
byte    *month2     :OUT     Month
byte    *day2       :OUT     Day
byte    *hour2      :OUT     Hour
byte    *minute2    :OUT     Minute
byte    *second2    :OUT     Second
```

[Return values] `The day of the week`

[Description] Gets the current date/time in the numeric format. Summer time (Daylight saving time) correction is provided.

- Date/Time functions -

[Function name] `LibGetDate`

[Syntax]

```
#include "define.h"
#include "libc.h"
#include "l_define.h"
#include "l_libc.h"
byte LibGetDate(byte *yearh, byte *yearl, byte *month, byte *day)
```

[Arguments]

```
byte      *yearh    :OUT      Year high
byte      *yearl    :OUT      Year low
byte      *month    :OUT      Month
byte      *day      :OUT      Day
```

[Return values] `The day of the week`

[Description] Gets the current date in the BCD format. No summer time correction.

- Date/Time functions -

[Function name] `LibGetTime`

[Syntax]
```
#include   "define.h"
#include   "libc.h"
#include   "l_define.h"
#include   "l_libc.h"
void LibGetTime(byte *hour, byte *minute, byte *second)
```

[Arguments]
```
byte   *hour      :OUT     Hour
byte   *minute    :OUT     Minute
byte   *second    :OUT     Second
```

[Return values] `None`

[Description] Gets the current time in the BCD format. No summer time correction.

- Date/Time functions -

[Function name] `LibGetDate2`

[Syntax]

```
#include  "define.h"
#include  "libc.h"
#include  "l_define.h"
#include  "l_libc.h"
byte LibGetDate2(word *year2, byte *month2, byte *day2)
```

[Arguments]

```
word    *year2     :OUT    Year
byte    *month2    :OUT    Month
byte    *day2      :OUT    Day
```

[Return values] `The day of the week`

[Description] Gets the current date in the numeric format. No summer time correction.

- Date/Time functions -

[Function name] `LibGetTime2`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibGetTime2(byte *hour2, byte *minute2, byte *second2)
```

[Arguments]

```
byte    *hour2      :OUT    Hour
byte    *minute2    :OUT    Minute
byte    *second2    :OUT    Second
```

[Return values] `None`

[Description] Gets the current time in the numeric format. No summer time correction.

- Date/Time functions -

[Function name] `LibAdjustTimeDeff2`

[Syntax]

```
     #include "define.h"
     #include "libc.h"
     #include "l_define.h"
     #include "l_libc.h"
     void LibAdjustTimeDeff2(int lag, word *year2, byte *month2, byte *day2, byte *dweek2,
byte *hour2, byte *minute2, byte *second2)
```

[Arguments]

int	lag	:IN	Time lag correction
word	*year2	:OUT	Year
byte	*month2	:OUT	Month
byte	*day2	:OUT	Day
byte	*dweek2	:OUT	Week
byte	*hour2	:OUT	Hour
byte	*minute2	:OUT	Minute
byte	*second2	:OUT	Second

[Return values] None

[Description] Corrects a date/time with the time lag specified by "lag".

- Date/Time functions -

[Function name] `LibChangeTotalDay`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
byte LibChangeTotalDay(word *year2, byte *month2, byte *day2, dword tday)
```

[Arguments]

word	*year2	:OUT	Year
byte	*month2	:OUT	Month
byte	*day2	:OUT	Day
dword	tday	:IN	Total number of days

[Return values] The day of the week

[Description] Converts the total number of days specified by "tday" into the date (numeric format).

[Examples of usage]

```
*s_dow = LibChangeTotalDay(&year,&month,&day,ttl_day);

/* Gets the date string for the flash search. */
LibNumToStr(&src_date[0],year,  4);
LibNumToStr(&src_date[4],month, 2);
LibNumToStr(&src_date[6],day,   2);
```

- Date/Time functions -

[Function name] `LibGetTotalDay2`

[Syntax]

```
#include  "define.h"
#include  "libc.h"
dword LibGetTotalDay2(word year2, byte month2, byte day2)
```

[Arguments]

```
word    year2      :IN     Year
byte    month2     :IN     Month
byte    day2       :IN     Day
```

[Return values] `Total number of days`

[Description] Gets the total number of days from the specified Year-Month-Day (numeric format).

[Examples of usage]

```
dword   ttl_day;
word    year;
byte    month,day,hour,minute,second;

LibGetDateTime2(&year,&month,&day,&hour,&minute,&second);
ttl_day = LibGetTotalDay2(year,month,day);
```

- Date/Time functions -

[Function name] `LibSetDateTime`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibSetDateTime(byte yearh, byte yearl, byte month, byte day,
                                          byte hour, byte minute)
```

[Arguments]

byte	yearh	:IN	Year high
byte	yearl	:IN	Year low
byte	month	:IN	Month
byte	day	:IN	Day
byte	hour	:IN	Hour
byte	minute	:IN	Minute

`All in BCD format`

[Return values] `None`

[Description] Updates the current date and time.

[Note] Be sure to specify the value with the summer time correction.

- Date/Time functions -

[Function name] `LibSetDateTime2`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     #include    "l_define.h"
     #include    "l_libc.h"
     void LibSetDateTime2(word year2, byte month2, byte day2, byte hour2, byte
minute2)
```

[Arguments]

```
     word    year2      :IN    Year
     byte    month2     :IN    Month
     byte    day2       :IN    Day
     byte    hour2      :IN    Hour
     byte    minute2    :IN    Minute

     All in numeric format.
```

[Return values] None

[Description] Updates the current date and time.

[Note] Be sure to specify the value with the summer time correction.

- Date/Time functions -

[Function name] `LibSetDate2`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
#include   "l_define.h"
#include   "l_libc.h"
void LibSetDate2(word year2, byte month2, byte day2)
```

[Arguments]

```
word    year2      :IN     Year
byte    month2     :IN     Month
byte    day2       :IN     Day

All in numeric format.
```

[Return values] `None`

[Description] Updates the current date.

[Note] Be sure to specify the value without the summer time correction.

- Date/Time functions -

[Function name] `LibSetTime2`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
#include   "l_define.h"
#include   "l_libc.h"
void LibSetTime2(byte hour2, byte minute2)
```

[Arguments]

```
byte    hour2      :IN     Hour
byte    minute2    :IN     Minute

All in numeric format.
```

[Return values] `None`

[Description] Updates the current time.

[Note] Be sure to specify the value without the summer time correction.

- Date/Time functions -

[Function name] `LibGetDow`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     byte LibGetDow(byte *buff)
```

[Arguments]

```
     byte    *buff      :IN     Date buffer     [8] ASCII format
```

[Return values]

```
The day of the week
0: Sun, 1: Mon, 2: Tue, 3: Wed, 4: Thu, 5: Fri, 6: Sat
```

[Description] Gets the day of the week from the date specified in "buff".

[Examples of usage]

```
    byte    date_buf[8];
    byte    dow;

    memcpy(date_buf,"19980401",8);
    dow = LibGetDow(date_buf);      /* Gets 3. */
```

- Date/Time functions -

[Function name] `LibGetDays`

[Syntax]

```
#include "define.h"
#include "libc.h"
byte LibGetDays(word *buf)
```

[Arguments]

```
word    *buff      :IN   Year-Month buffer  [2] Numeric format
```

[Return values] `Number of days`

[Description] Gets the number of days of the month from the year and month specified in "buff".

[Examples of usage]

```
word    ym[2];
byte    days;

ym[0]   = 1996;
ym[1]   = 2;

days = LibGetDays(ym);  /* Gets 29. */
```

- Date/Time functions -

[Function name] `LibChkFuture`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
byte LibChkFuture(byte *tgt, byte *now, byte ct)
```

[Arguments]

```
byte    *tgt       :Compare to date/time  (Target)
byte    *now       :Compare from date/time  (Source)
byte    ct         :Comparison method    8:Date
                                         4: Time
```

[Return values]

```
Comparison result   0: Compared to the past.
                    1: Compared to the present.
                    2: Compared to the future.
```

[Description] Performs the size comparison (old and new comparison) of the date/time data.

- Date/Time functions -

[Function name] `LibSummerTimeSet`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
void LibSummerTimeSet(void)
```

[Arguments] None

[Return values] None

[Description] Set summer time for Home and World.
This is created to notify the item set in a system variable to the BIOS side.

- Date/Time functions -

[Function name] `LibDateDisp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibDateDisp(byte *buff, byte type, int xp, int yp, int xep,
                                  byte dmode, byte F_Type)
```

[Arguments]

```
byte    *buff      :IN     Date string buffer  [8] ASCII FORMAT
byte    type       :IN     Day of the week display: preset/non
                             WEEK_ON: Present
                             WEEK_OFF: None
int      xp        :IN     Display start coordinate Horizontal
int      yp        :IN     Display start coordinate Vertical
int      xep       :IN     Display end coordinate Horizontal
byte     dmode     :IN     Display format
                             IN_MODE: For input
                             DISP_MODE: For display
                             DISP_YM: Year-Month
                             DISP_MY: Month-Year
                             IN_YM: Year-Month input
byte     F_Type    :IN     Font type
```

[Return values]

```
Execution result  TRUE: Normal
                  FALSE: Abnormal (buff[]: All "0".)
```

[Description] Displays the date string specified by "buff[]" to the specified coordinates.

- Date/Time functions -

[Function name] `LibWait`

[Syntax]

```
     #include   "define.h"
     #include   "libc.h"
     void LibWait(byte time)
```

[Arguments]

```
     byte      time      :IN     Wait time
                                   IB_125MWAIT:     125 msec.
                                   IB_250MWAIT:     250 msec.
                                   IB_500MWAIT:     500 msec.
                                   IB_1SWAIT:       1 sec.
```

[Return values] `None`

[Description] Makes the dummy wait for the specified period of time.

- Date/Time functions -

[Function name] `LibCheckDate`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
bool LibCheckDate(byte *dbuff)
```

[Arguments]

```
byte     *dbuff    :IN     Date type string [8]    (YYYYMMDD)
```

[Return values] `bool  Inspection result`

[Description] Checks if the character string specified by "dbuff" is valid as the date type.

[Examples of usage]

```
byte    date_buf[8];
bool    chk;

memcpy(date_buf,"19980229",8);

chk = LibCheckDate(date_buf);   /* Result is FALSE.*/
```

- Date/Time functions -

[Function name] `LibChkTimeBuf`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
bool LibChkTimeBuf(byte far *buf)
```

[Arguments]

```
byte far  *buf       :IN     Time data buffer  [5] "HHMMA"
```

[Return values]

```
bool  Inspection result  TRUE: Normal
                         FALSE: Abnormal
```

[Description] Performs the validity test to the time format data specified in "buf".

[Examples of usage]

```
byte    t_buf[5];

memcpy(t_buf,"1345P",5);

if(LibChkTimeBuf(t_buf)==TRUE){
        •
        •
        •
```

- Date/Time functions -

[Function name] `LibClkDispLine`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
void LibClkDispLine(LPTIMEKEYBCTRL lptbl)
```

[Arguments]

```
LPTIMEKEYBCTRL  lptbl       :IN     Time array
```

[Return values] `None`

[Description] Displays the contents of the specified time input structure lptbl(LPTIMEKEYBCTRL).

- Date/Time functions -

[Function name] `LibClkDispCursor`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
void LibClkDispCursor(LPTIMEKEYBCTRL lptbl,int csrpos)
```

[Arguments]

```
LPTIMEKEYBCTRL  lptbl       :IN     Time array
int      csrpos             :IN     Cursor position
```

[Return values] `None`

[Description] Displays the cursor of the specified time input structure lptbl(LPTIMEKEYBCTRL). The character string must be displayed using LibClkDispLine before this function is applied. (Because this function only inverts the specified part of the characters internally.)

- Date/Time functions -

[Function name] `LibConvRaw2Lib`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
void LibConvRaw2Lib(byte *dest,byte *src,bool IsLarge)
```

[Arguments]

```
byte    *dest      :IN     4-byte time (HHMM) before conversion.
byte    *src       :OUT    Converted library time (HHMM).
bool    IsLarge    :IN     AM/PM in capital
                          TRUE: AM/PM in capital after conversion.
```

[Return values] `None`

[Description] Converts the "HHMM" array for 24-hour system to the acceptable value (Library time) with LibInputTime based on the system settings.

- Date/Time functions -

[Function name] `LibConvRaw2Lib2`

[Syntax]

```
     #include   "define.h"
     #include   "libc.h"
     void LibConvRaw2Lib(byte *dest,byte *src,bool IsLarge)
```

[Arguments]

```
     byte    *src       :IN      4 byte time before conversion ('HHMM')
     byte    *dest      :OUT     Conversion time ('HH:MMA')
     bool    IsLarge    :IN      Specify uppercase for AM/PM
                             TRUE: Uppercase letters 'AM/PM' after conversion
```

[Return values] None

[Description] This operates the same as LibConvRaw2Lib except that it inserts ":" between hours and minutes when outputting the time.

- Date/Time functions -

[Function name] `LibConvLib2Raw`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    void LibConvLib2Raw(byte *dest,byte *src)
```

[Arguments]

```
    byte    *dest      :IN     Library time before conversion ('HHMMA')
    byte    *src       :OUT    4-byte time after conversion ('HHMM')
```

[Return values] `None`

[Description] Converts the acceptable value (Library time) of LibInputTime to the 4-byte character string for 24-hour system based on the system settings.

- Date/Time functions -

[Function name] `LibGetCursorPos`

[Syntax]

```
#include "define.h"
#include "libc.h"
int LibGetCursorPos(int x,int y,LPTIMEKEYBCTRL lptbl)
```

[Arguments]

```
int       x                  :IN     X-coordinate of touch position
int       y                  :IN     Y-coordinate of touch position
LPTIMEKEYBCTRL     lptbl     :IN     Time array
```

[Return values] `int   Cursor position`

[Description] Gets the initial value of the time-input cursor from the touch position.

- Date/Time functions -

[Function name] `LibJumpDate`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
bool LibJumpDate(byte *s_date)
```

[Arguments]

```
byte    *s_date    :IN/OUT Date buffer     [8]
```

[Return values]

```
bool   Execution result
                TRUE:  Rewrite
                FALSE: None
                HALF:  Schedule hardware-icon is touched.
```

[Description] Displays the date jump screen and changes the contents in the buffer specified by "s_date[]" using the Calendar keyboard.

- Character input/Drag event functions -

[Function name] `LibTxtInit`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
void LibTxtInit(TXTP *tp)
```

[Arguments]
```
TXTP     *tp      :IN/OUT    Text input information
```

[Return values] None

[Description] Initializes various variables for text input.

[Examples of usage]
```
TXTP    m_in_tp;

word telgd[] = {    213,    /* NAME     */
           202,    /* ADDRESS_(H) */
           203,    /* FAX_(B)     */
           204,    /* PHONE_(B)   */
           205,    /* E-MAIL      */
           206,    /* EMPLOYER    */
           207,    /* FAX_(H)     */
       };

m_in_tp.st_x    = M_ST_X; /* Start coordinate (X) of text display  */
m_in_tp.st_y    = M_ST_Y; /* Start coordinate (Y) of text display  */
m_in_tp.ed_x    = M_ED_X; /* End coordinate (X) of text display  */
m_in_tp.it_y    = M_IT_Y; /* Text display line spacing (Y)   */
m_in_tp.MAXGYO  = M_MAXG; /* Number of text display lines    */
m_in_tp.font    = IB_PFONT1;  /* Display font type        */
m_in_tp.csen    = TRUE;       /* Cursor display disabled */
m_in_tp.rtnen   = TRUE;       /* CR code display disabled */
m_in_tp.maxmj   = 2048;       /* Maximum number of input characters */
m_in_tp.txbf    = mtxbf;      /* Text buffer address specification  */
mtxbf[0]         = 0;         /* Initialization of text buffer */
m_in_tp.gdcmt   = telgd;      /* Guidance comment table     */
m_in_tp.txtobj  = OBJ_PAY_DAT; /* Text area object code */
m_in_tp.sbrobj  = OBJ_SCR_BAR; /* Scroll bar object code */

LibTxtInit(&m_in_tp);   /* Initialization of text input */
```

- Character input/Drag event functions -

[Function name] `LibTxtTchSet`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibTxtTchSet(TXTP *tp)
```

[Arguments]

```
     TCHSTS      *tsts           :IN     Touch status information
```

[Return values] `None`

[Description] Registers the touch table for text input.
This function is used in the preparation phase for text input.

- Character input/Drag event functions -

[Function name] `LibTxtInp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibTxtInp(byte keycd, TCHSTS *tsts, TXTP *tp)
```

[Arguments]

```
byte      keycd     :IN      Character code  (a value from keyboard)
TCHSTS    *tsts     :OUT    Touch status information
TXTP      *tp       :IN/OUT     Text input information
```

[Return values] None

[Description] Controls the text input.
This function uses the internal touch waiting to perform the character input process by software keyboard and performs the drag selection process.

[Examples of usage]

```
/*** Data input loop ***/
while(mem_st==NEW_INP){
    if(LibTxtDsp(&m_in_tp)==TRUE);      /* Display during text input  */
        LibPutDisp();

    kycd = LibGetKeyM(&tsts);  /*  Software key wait */

    if(kycd==KEY_NONE){

/***Process for touching a portion other than the text area, software key, and scroll bar.***/
    if(tsts.obj==OBJ_IC_DATAKEY){/*When touching "Data display" icon.*/
   if(LibIconClick(&TicnDataKey,&tsts)==TRUE){
      mem_st = DATA_DSP;/*Escape from new input loop to data display.*/
      LibTchInit();
   }

 /*** Copy, cut, and paste process  ***/
        }else if(tsts.obj == OBJ_HIC_CONT){
   mm = LibCpMenu();    /* Copy & paste menu  (for DEBUG) */
   switch(mm){
       case 0:
           m_in_tp.txtst = TXTCUT;      /* Cut */
           break;
       case 1:
           m_in_tp.txtst = TXTCOPY;    /* Copy */
```

```
            break;
        case 2:
            m_in_tp.txtst = TXTPASTE;   /* Paste */
            break;
    }
          }
      }

 /*** Main process of text input ***/
     LibTxtInp(kycd,&tsts,&m_in_tp);
}
```

- Character input/Drag event functions -

[Function name] `LibTxtDsp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibTxtDsp(TXTP *tp)
```

[Arguments]

```
TXTP      *tp      :IN/OUT     Text input information
```

[Return values] Screen update status. TRUE: Present
FALSE: None

[Description] Updates the display contents during text input.
This function also displays the software keyboard in addition to display of characters already input.

[Examples of usage]

```
/*** Data input loop ***/
while(mem_st==NEW_INP){
    if(LibTxtDsp(&m_in_tp)==TRUE);  /* Display during text input  */
    LibPutDisp();
    •
    •
    •
```

- Character input/Drag event functions -

[Function name] `LibTxtDspC`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibTxtDspC(TXTP *tp)
```

[Arguments]

```
TXTP     *tp      :IN/OUT    Text input information
```

[Return values]

```
Screen update status.        TRUE: Present
                             FALSE:  None
```

[Description] Updates the display contents during data display.

[Examples of usage]

```
/***Data display loop  ***/
while(mem_st==DATA_DSP){
    if(LibTxtDspC(&m_dd_tp)==TRUE)      /* Text display  */
    LibPutDisp();
    •
    •
    •
```

- Character input/Drag event functions -

[Function name] `LibTxtDspInit`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    void LibTxtDspInit(TXTP *tp)
```

[Arguments]

```
    TXTP     *tp      :IN/OUT     Text input information
```

[Return values] `None`

[Description] Initializes various variables for data display.

[Examples of usage]

```
  /** Text input setting **/
  m_dd_tp.st_x = M_ST_X;    /* Start coordinate (X) of text display */
  m_dd_tp.st_y = M_ST_Y;    /* Start coordinate (Y) of text display */
  m_dd_tp.ed_x = M_ED_X;    /* End coordinate (X) of text display */
  m_dd_tp.it_y = M_IT_Y;    /* Text display line spacing (Y)  */
  m_dd_tp.MAXGYO = M_MAXG;    /* Number of text display lines    */
  m_dd_tp.font = IB_PFONT1;  /* Display font type  */
  m_dd_tp.csen = TRUE;  /* Cursor display disabled  */
  m_dd_tp.rtnen = FALSE;    /* CR code display disabled  */
  m_dd_tp.maxmj = 100; /*Maximum number of allowable input characters*/
  m_dd_tp.txbf = mtxbf;  /* Text buffer address specification */

  LibTxtDspInit(&m_dd_tp);  /* Initialization of text input  */
```

- Character input/Drag event functions -

[Function name] `LibTxtDspS`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibTxtDspS(TXTP *tp, TCHSTS *tsts)
```

[Arguments]

```
TXTP      *tp       :IN/OUT   Text input information
TCHSTS    *tsts     :OUT      Touch status information
```

[Return values] None

[Description] Controls events during data display.

[Examples of usage]

```
while(1){
    LibTchWait(&tsts);

    switch(tsts.obj){
    case OBJ_HEAD00:    /* Screen Shot */
   break;

        case OBJ_HEAD01:    /* New */
   break;

        case OBJ_HIC_ESC:

        default:
   break;
    }
    LibTxtDspS(&EventPrm,&tsts);
}
```

- Character input/Drag event functions -

[Function name] `LibGetCursor`

[Syntax]
```
     #include    "define.h"
     #include    "libc.h"
     bool LibGetCursor(int *c_xp, int *c_yp, int *c_xsize ,int *c_ysize)
```

[Arguments]
```
     int      *c_xp             :OUT    Coordinate - Horizontal
     int      *c_yp             :OUT    Coordinate - Vertical
     int      *c_xsize          :OUT    Horizontal size
     int      *c_ysize          :OUT    Vertical size
```

[Return values]
```
bool  Blink/No blink  TRUE: ON   Being ON with LibCurBlnkOn().
                      HALF: ON   Being ON with LibCurBlnkOn2().
                      FALSE: OFF
```

[Description] Gets the cursor status.

- Character input/Drag event functions -

[Function name] `LibCurBlnkOn`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCurBlnkOn(int x, int y, int xsize, int ysize)
```

[Arguments]

```
int      x        :IN    Coordinate - Horizontal
int      y        :IN    Coordinate - Vertical
int      xsize    :IN    Horizontal size
int      ysize    :IN    Vertical size
```

[Return values] `None`

[Description] Blinks a cursor.
The cursor shape is a reverse of the specified area.

- Character input/Drag event functions -

[Function name] `LibCurBlnkOn2`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibCurBlnkOn2(int x, int y, int xsize, int ysize)
```

[Arguments]

```
     int       x         :IN     Coordinate - Horizontal
     int       y         :IN     Coordinate - Vertical
     int       xsize     :IN     Horizontal size
     int       ysize     :IN     Vertical size
```

[Return values] None

[Description] Blinks a cursor.
The cursor shape is a reverse of the blank part of the specified area.

- Character input/Drag event functions -

[Function name] `LibCurBlnkOff`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibCurBlnkOff(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Turns off a cursor.

- Character input/Drag event functions -

[Function name] `LibCurErase`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibCurErase(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Clears a cursor.
Puts a cursor to the off state unconditionally.

- Character input/Drag event functions -

[Function name] `LibTxtKeyWordSet`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibTxtKeyWordSet(TXTP *tp)
```

[Arguments]

```
TXTP      *tp       :IN/OUT     Text input information
```

[Return values] `None`

[Description] Performs the keyword registration for text input.
This function is the internal processing of the character input library. However, this function is executed to register the word handled last as the keyword when ending the input process by pressing the SET or ESC button.

- Character input/Drag event functions -

[Function name] `LibTxtWrapSw`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibTxtWrapSw(int Sw)
```

[Arguments]

```
int Sw    :ON       Word wrap enabled
          :OFF      Word wrap disabled
```

[Return values] `None`

[Description] Control to enable/disable word wrap in text processing by Sw.

- Message functions -

[Function name] `LibPutMessage`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibPutMessage(word no,int p_x,int p_y,byte type)
```

[Arguments]

```
word      no        :IN     Message number
int       p_x       :IN     Coordinate - Horizontal
int       p_y       :IN     Coordinate - Vertical
byte      type      :IN     Font type
                              IB_PFONT1: Data type
                              IB_PFONT2: Background type
                              IB_PFONT3: For title
                              IB_CG57FONT: 5*7
```

[Return values] `None`

[Description] Displays a built-in 5-language message corresponding to a number specified by "no" with a font type specified by "type" at specified coordinates.

[Examples of usage]

```
LibPutMessage(57,20,4,IB_PFONT2);
LibPutMessage(71,86,4,IB_PFONT2);
```

- Message functions -

[Function name] `LibPutMessageCenter`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibPutMessageCenter(word no,int p_x1,int p_x2,int p_y,byte type)
```

[Arguments]

```
word     no       :IN     Message number
int      p_x1     :IN     Left edge coordinate (start)
int      p_x2     :IN     Right edge coordinate (end)
int      p_y      :IN     Ordinates
byte     type     :IN     Font type
                           IB_PFONT1: Data type
                           IB_PFONT2: Background type
                           IB_PFONT3: For title
                           IB_CG57FONT: 5*7
```

[Return values] None

[Description] Displays a built-in 5-language message corresponding to a number specified by "no" with a font type specified by "type" at a position between two X-coordinates (p_x1, p_x2) so that it is centered.

[Note] If the length of the message string is too long to fit to the space between two X-coordinates, the message will not be displayed. Remember that the length of character string is changed by the language selection.

[Examples of usage]

```
  case 0:
      LibPutMessageCenter(306, 23, 291,43,IB_PFONT2);/* SCHEDULE ALARM (TITLE) */
      break;
  case 1:
      LibPutMessageCenter(307, 23, 291,43,IB_PFONT2);/* REMINDAR ALARM (TITLE) */
      break;
  case 2:
      LibPutMessageCenter(308, 23, 291,43,IB_PFONT2);/* TODO ALARM (TITLE) */
      break;
```

- Message functions -

[Function name] `LibPutMessageCenter2`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
void LibPutMessageCenter2(word no,int p_x,int p_y,byte type)
```

[Arguments]
```
word      no                 :IN     Message number
int       p_x       :IN     Coordinate - Horizontal
int       p_y       :IN     Coordinate - Vertical
byte      type      :IN     Font type
                              IB_PFONT1: Data type
                              IB_PFONT2: Background type
                              IB_PFONT3: For title
                              IB_CG57FONT: 5*7
```
[Return values] None

[Description] Displays a built-in 5-language message corresponding to a number specified by "no" with a font type specified by "type" so that the coordinate position (p_x, p_y) is located at the center of the message.

[Examples of usage]
```
LibPutMessageCenter2(300,70,13,IB_PFONT2);
```

- Message functions -

[Function name] `LibPutMessageRight`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibPutMessageRight(word no,int p_x,int p_y,byte type)
```

[Arguments]

```
word     no       :IN     Message number
int      p_x      :IN     Coordinate - Horizontal
int      p_y      :IN     Coordinate - Vertical
byte     type     :IN     Font type
                            IB_PFONT1: Data type
                            IB_PFONT2: Background type
                            IB_PFONT3: For title
                            IB_CG57FONT: 5*7
```

[Return values] None

[Description] Displays a built-in 5-language message corresponding to a number specified by "no" with a font type specified by "type" so that it is right justified corresponding to the coordinates (p_x, p_y).

[Examples of usage]

```
LibPutMessageRight(228,TRN_DAT_COL-5,TrnDatRow[DATE]+1,      IB_PFONT2);
LibPutMessageRight(109,TRN_DAT_COL-5,TrnDatRow[PAYEE]+1,     IB_PFONT2);
LibPutMessageRight( 77,TRN_DAT_COL-5,TrnDatRow[AMOUNT]+1,    IB_PFONT2);
LibPutMessageRight( 86,TRN_DAT_COL-5,TrnDatRow[CHECK_NUM]+1,IB_PFONT2);
LibPutMessageRight( 76,TRN_DAT_COL-5,TrnDatRow[CATEGORY]+1, IB_PFONT2);
```

- Message functions -

[Function name] `LibReadMessage`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibReadMessage(word no,byte *buf)
```

[Arguments]

```
word    no                  :IN     Message number
byte    *buf       :OUT    Character string buffer
```

[Return values] `None`

[Description] Reads a built-in 5-language message character string corresponding to a number specified by "no" and writes it into the buffer "buf".

[Note] This function does not check if there is enough space to write the character string. So, it is necessary to allocate enough space to "buf".

[Examples of usage]

```
byte    type_str[30];

LibReadMessage(102,type_str);
```

- Message functions -

[Function name] `LibGetMessCnt`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
byte LibGetMessCnt(int mes_no)
```

[Arguments]

```
int      mes_no    :IN     Message number
```

[Return values] `byte    Number of message lines`

[Description] Gets the number of lines of the built-in 5-language message.

- Message functions -

[Function name] `LibDspWinMessage`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibDspWinMessage(byte g_no,int mes_no,byte b_cnt,byte b_type)
```

[Arguments]

```
byte     g_no      :IN     Icon (graphic) number
                   None when IB_MWIN_NO_ICON
int      mes_no    :IN     Message number
byte     b_cnt     :IN     Number of buttons (0 - 2)
byte     b_type    :IN     Button type
                          IB_MWIN_NONE: None
                          IB_MWIN_YES_NO: YES/NO (Two buttons)
                          IB_MWIN_SET_ESC: SET/ESC (Two buttons)
                          IB_MWIN_OK: OK (One button)
                          IB_MWIN_SET: SET (One button)
                          IB_MWIN_ESC: ESC (One button)

                          IX_MWIN_CENTER:  Assign window position center.
```

[Return values] None

[Description] Displays a dialog message.
The window position is centered in the screen when IX_MWIN_CENTER is logical ORed with "b_type".

- Message functions -

[Function name] `LibGetWinMessSize`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibGetWinMessSize(int mes_no,byte w_pos,byte b_cnt,int *y,int *y_size)
```

[Arguments]

```
int      mes_no   :IN     Message number
byte     w_pos    :in     Window display position
                            0x00:     Bottom of screen
                            IX_MWIN_CENTER: Center of screen
byte     b_cnt    :IN     Number of buttons(0 -2)
int      *y       :OUT    Window abscissas
int      *y_size  :OUT    Window vertical size
```

[Return values] None

[Description] Gets the window position and size for the message dialog.

- Message functions -

[Function name] `LibErrorDisp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibErrorDisp(word err_code)
```

[Arguments]

```
word    err_code   :IN     Error code
```

[Return values] `None`

[Description] Displays a message that corresponds to FLASH memory error code.

[Note] Normally, specify the global variable FlashStatus to "err_code". Additionally, the execution timing has to be immediately after FLASH access. Every time when the FLASH related BIOS is executed, the error code is output to "FlashStatus". So it is necessary to update it to the latest value.

[Examples of usage]

```
f_handle = LibFileWrite(&FileBuf,&FileInf); /* WRITES IN THE FLASH MEMORY. */

if(f_handle == TRUE){
    /* To normal process */
}
else{
    LibErrorDisp(FlashStatus);
}
```

- Character string functions -

[Function name] `LibBCD2Ascii`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibBCD2Ascii(byte bcd, byte *ascii)
```

[Arguments]

```
     byte    bcd        :IN     BCD code
     byte    *ascii     :OUT    Buffer for ASCII code
```

[Return values] `None`

[Description] Converts a 1-byte BCD code specified by "bcd" into 2-byte ASCII code, and writes it to "ascii".

- Character string functions -

[Function name] `LibAscii2BCD`

[Syntax]
```
     #include    "define.h"
     #include    "libc.h"
     byte LibAscii2BCD(char *ascii)
```

[Arguments]
```
     char    *ascii     :IN     Buffer for ASCII code
```

[Return values] `BCD code after conversion`

[Description] Converts a 2-byte ASCII code specified by "ascii" into 1-byte BCD code, and returns it.

- Character string functions -

[Function name] `LibNumoStr`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibNumToStr(byte *buff, word target, byte j)
```

[Arguments]

byte	*buff	:OUT	Converts to a string
word	target	:IN	Converts a number
byte	j	:IN	Digit

[Return values] None

[Description] Converts a numeric number specified by "target" into character string.

[Note] A NULL is not set to the end of the character string.

- Character string functions -

[Function name] `LibStoNum`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibSToNum(byte *buff, word *res_num, byte j)
```

[Arguments]

byte	*buff	:IN	Convert a string
word	*res_num	:OUT	Convert to a number
byte	j	:IN	Digit

[Return values] None

[Description] Converts a character string specified by "buff" into numeric number.

- Character string functions -

[Function name] `LibCuttextRtn`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibCutTextRtn(byte *txbf)
```

[Arguments]

```
byte    *txbf      :IN/OUT     Text buffer
```

[Return values]

```
Text state         TRUE: Has data in the buffer.
                   FALSE: None
```

[Description] Deletes the CR code at the end of a text or an item.

- Character string functions -

[Function name] `LibKeyWordInit`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibKeyWordInit(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Initializes the keyword registration area.

- Character string functions -

[Function name] `LibKeyWordSet`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibKeyWordSet(byte *key_str)
```

[Arguments]

```
byte    *key_str   :IN     Keyword string
```

[Return values] `None`

[Description] Registers a character string specified by “key_str” in the keyword area. At this time, if the internal area is full, data is deleted from the oldest one automatically.

- Character string functions -

[Function name] `LibKeyWordFSrch`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibKeyWordFSrch(byte *srch_str,byte *key_str)
```

[Arguments]

byte	*srch_str	:IN	String buffer for search
byte	*key_str	:OUT	String buffer for search result

[Return values]

Execution result	TRUE: Match data presents.
	FALSE: None

[Description] Performs the first search to find a character string specified by “srch_str”. If the relevant keyword is found, this function writes it to “key_str”. If no matching keyword is found, this function writes NULL to the start address of “key_str”.

[Examples of usage]

```
bool    ans;

LibKeyWordInit();

LibKeyWordSet("apple");     /* 0 */
LibKeyWordSet("and");       /* 1 */
LibKeyWordSet("able");      /* 2 */
LibKeyWordSet("again");     /* 3 */
LibKeyWordSet("against");   /* 4 */
LibKeyWordSet("address");   /* 5 */
LibKeyWordSet("beer");      /* 6 */
LibKeyWordSet("black");     /* 7 */

ans = LibKeyWordFSrch("ag",key_str);    /* Data 4 matched. */
```

- Character string functions -

[Function name] `LibKeyWordNSrch`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
bool LibKeyWordNSrch(byte *srch_str,byte *key_str)
```

[Arguments]
```
byte    *srch_str  :IN     String buffer for search
byte    *key_str   :OUT    String buffer for search result
```

[Return values]
```
Execution result    TRUE: Has match data.
                    FALSE: No match data.
```

[Description] Performs the next search to find a character string specified by “srch_str”.
If the matching keyword is found, this function writes the character into “key_str”.
This function writes NULL at the start address of "key_str" if no matching keyword is found.

[Note] Call this after executing the first search LibKeyWordFSrch().

[Examples of usage]
```
bool    ans;

LibKeyWordInit();

LibKeyWordSet("apple");      /* 0 */
LibKeyWordSet("and");        /* 1 */
LibKeyWordSet("able");       /* 2 */
LibKeyWordSet("again");      /* 3 */
LibKeyWordSet("against");    /* 4 */
LibKeyWordSet("address");    /* 5 */
LibKeyWordSet("beer");       /* 6 */
LibKeyWordSet("black");      /* 7 */

ans = LibKeyWordFSrch("ag",key_str);    /* Data 4 matched. */
ans = LibKeyWordNSrch("ag",key_str);    /* Data 3 matched.*/
```

- Character string functions -

[Function name] `LibKeyWordSrchSub`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibKeyWordSrchSub(byte sw,byte *srch_str,byte *key_str)
```

[Arguments]

```
byte     sw                :IN     Search switch
                                IB_KEYWD_FSRCH: First search
                                IB_KEYWD_NSRCH: NEXT
byte     *srch_str         :IN     String buffer for search
byte     *key_str          :OUT    String buffer for search result
```

[Return values]

```
Execution result     TRUE: Has match data.
                     FALSE: No match data.
```

[Description] Performs a search with the search type specified by "sw" to find the character string specified by "srch_str". If the matching keyword is found, this function writes the character string into “key_str”. This function writes NULL at the start address of "key_str" if no matching keyword is found.

- Character string functions -

[Function name] `LibChangeBcdVal`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
byte LibChangeBcdVal(byte bcd)
```

[Arguments]
```
byte    bcd        :IN     BCD code
```

[Return values] `Numeric number`

[Description] Converts a BCD code specified by "bcd" into numeric number.

- Character string functions -

[Function name] `LibChangeValBcd`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     byte LibChangeValBcd(byte val)
```

[Arguments]

```
     byte    val        :IN     Numeric number
```

[Return values] `BCD code`

[Description] Converts a numeric number specified by "val" into BCD code.

- Character string functions -

[Function name] `LibLblAreaWrite`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibLblAreaWrite(byte *fb, byte typ)
```

[Arguments]

```
byte    *fb        :IN     Label content buffer
byte    typ        :IN     Types
                             00H to 04H = Contacts: 197-Byte/Block
                             05H to 09H = Memo: 15-Byte/Block
```

[Return values] `None`

[Description] Registers the label contents of the CONTACTS mode.

- Character string functions -

[Function name] `LibLblAreaRead`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    void LibLblAreaRead(byte *fb, byte typ)
```

[Arguments]

```
    byte    *fb        :OUT    Label content buffer
    byte    typ        :IN     Type
                           00H to 04H = Contacts: 197-Byte/Block
                           05H to 09H = Memo: 15-Byte/Block
```

[Return values] `None`

[Description] Gets the registered label contents of the CONTACTS mode.

- Character string functions -

[Function name] `LibLblAreaClr`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibLblAreaClr(void)
```

[Arguments] None

[Return values] None

[Description] Clear the save area of label names (category and item names) in the CONTACTS mode.

- Handwriting (INK) functions -

[Function name] `LibDrawInit`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibDrawInit(INIT_PACS draw_prm)
```

[Arguments]

```
INIT_PACS   draw_prm         :IN     Parameter table
```

[Return values] `None`

[Description] Initializes the drawing BIOS.

- Handwriting (INK) functions -

[Function name] `LibDrawSetPtn`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibDrawSetPtn(byte val)
```

[Arguments]

```
byte    val        :IN     Contrast    (0 - 10)
```

[Return values] `None`

[Description] Specifies a contrast of the handwriting pen.

- Handwriting (INK) functions -

[Function name] `LibDrawSetClipArea`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibDrawSetClipArea(SETCLIPAREA_PACS draw_prm)
```

[Arguments]

```
SETCLIPAREA_PACS    draw_prm      :IN     Parameter table
```

[Return values] `None`

[Description] Specifies a drawing area.

- Handwriting (INK) functions -

[Function name] `LibDrawSetPoint`

[Syntax]
```
     #include    "define.h"
     #include    "libc.h"
     void LibDrawSetPoint(byte page,int x,int y,byte point,byte mode)
```

[Arguments]
```
     byte      page      :IN    Write page   0:Real screen
                                             1:Background screen + VRAM/DD
                                          * When no background: Always VRAM+DD.
     int       x         :IN     Draw x-coordinate
     int       y         :IN     Draw y-coordinate
     byte      point     :IN     Pen size   0: 1-dot pen
                                          1: 2-dot pen
                                          2: 4-dot pen

     byte    mode        :IN     Drawing mode
```

[Return values] `None`

[Description] Draws a dot.

- Handwriting (INK) functions -

[Function name] `LibDrawLine`

[Syntax]
```
     #include    "define.h"
     #include    "libc.h"
     void LibDrawLine(DRAWLINE_PACS draw_prm,byte page)
```

[Arguments]
```
     DRAWLINE_PACS draw_prm  :IN  Parameter table
     byte     page           :IN  Write page  0:Real screen
                                              1:Background screen + VRAM/DD
                            * When no background: Always VRAM+DD.
```
[Return values] `None`

[Description] Draws a line.

- Handwriting (INK) functions -

[Function name] `LibDrawBox`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibDrawBox(DRAWBOX_PACS draw_prm,byte page)
```

[Arguments]

```
     RAWBOX_PACS    draw_prm :IN  Parameter table
     byte      page          :IN  Write page  0:Real screen
                                              1:Background screen + VRAM/DD
                             * When no background: Always VRAM+DD.
```

[Return values] `None`

[Description] Draws a box.

- Handwriting (INK) functions -

[Function name] `LibDrawCircle`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibDrawCircle(DRAWCIRCLE_PACS draw_prm,byte page)
```

[Arguments]

```
DRAWCIRCLE_PACS draw_prm  :IN  Parameter table
byte      page            :IN  Write page 0:Real screen
                                          1:Background screen + VRAM/DD
                                 * When no background: Always VRAM+DD.
```

[Return values] `None`

[Description] Draws a circle.

- Handwriting (INK) functions -

[Function name] `LibDrawFillArea`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibDrawFillArea(FillArea_PACS draw_prm,byte page)
```

[Arguments]

```
illArea_PACS   draw_prm      :IN     Parameter table
byte     page                :IN     Write page/contrast
```

[Return values] `None`

[Description] Fills the rectangular area

- Handwriting (INK) functions -

[Function name] `LibDrawTransDD`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibDrawTransDD(TransDD_PACS draw_prm,byte dd)
```

[Arguments]

```
TransDD_PACS    draw_prm     :IN     Parameter table
byte      dd                 :IN     Transfer zone
                                  0:VRAM -> Background
                                  1:VRAM -> Raw
                                  2:Raw -> Background
                                  3:Background -> Raw
                                  4:VRAM -> DD
```

[Return values] `None`

[Description] Transfers the specified VRAM area to other VRAM defined individually.

- Handwriting (INK) functions -

[Function name] `LibDrawTransAll`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibDrawTransAll(byte val)
```

[Arguments]

```
byte    val        :IN     Transfer zone
                         * 0:VRAM -> Raw
                         * 1:VRAM -> BackGround
                           2:VRAM -> VRAM  ;;;;;;(NOP)
                           3:VRAM -> DD (PutDisp)
                         * 4:Clear Raw, BG  -> VRAM(DD)
                         * 5:Clear BG, Raw -> VRAM(DD)
                         * 6:BackGround + Raw -> VRAM(DD) (Full-PutDisp)
                           7:(Clear BG & Raw &) VRAM CLEAR  (ALL CLEAR)

     The item with * symbol is invalid (NOP) in the no background mode.
```

[Return values] `None`

[Description] Transfers the entire screen data between the specified virtual VRAMs, and between the system VRAMs.

- Handwriting (INK) functions -

[Function name] `LibDrawPutImage`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibDrawPutImage(PutImage_PACS draw_prm,byte page)
```

[Arguments]

```
PutImage_PACS   draw_prm :IN  Parameter table
byte     page            :IN  Write page  0:Real screen
                                          1:Background screen + VRAM/DD
                         * When no background: Always VRAM+DD.
```

[Return values] `None`

[Description] Writes an image to VRAM.

- Handwriting (INK) functions -

[Function name] `LibDrawGetImage`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
bool LibDrawGetImage(GetImage_PACS draw_prm,byte page)
```

[Arguments]
```
GetImage_PACS   draw_prm  :IN  Parameter table
byte     page             :IN  Read page     0:Real screen
                                             1:Background screen
                                             2:VRAM
                                   * When no background: Always VRAM
```

[Return values]
```
Execution result   TRUE: Succeeded
                   FALSE: Failed
```

[Description] Gets an image from VRAM.

- Handwriting (INK) functions -

[Function name] `LibDrawReductImage`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibDrawReductImage(ReductImage_PACS draw_prm)
```

[Arguments]

```
ReductImage_PACS    draw_prm   :IN     Parameter table
```

[Return values]

```
Execution result     TRUE: Succeeded
                     FALSE: Failed
```

[Description] Reduces an image.

- Handwriting (INK) functions -

[Function name] `LibDrawPrmCall`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     #include    "l_define.h"
     #include    "l_libc.h"
     int LibDrawPrmCall(byte func_no,void *draw_prm,byte al)
```

[Arguments]

```
     byte    func_no    :IN     Drawing BIOS Function number
     void    *draw_prm  :IN     Parameter table address
     byte    al                   :IN     AL Register input value
```

[Return values] `ax register output value`

[Description] Calls a drawing BIOS using the function number specified by "func_no".

- Handwriting (INK) functions -

[Function name] `LibScrShot`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibScrShot(SHOT_INF *s_inf)
```

[Arguments]

```
SHOT_INF    *s_inf :IN/OUT     Image information buffer
```

[Return values]

```
bool   Execution result     TRUE:  Succeeded
                            FALSE:  Failed
```

[Description] Executes the screen-shot process.
This function gets VRAM data from the image information specified by “s_inf” and transfers it to the handwriting mode started up in the dual-window. After completion of the handwriting mode, it returns to the caller.

[Note] "s_inf" must be a global variable pointer. It should not be a local variable. This function cannot be used during mode operation in the dual-window. (Because the handwriting mode is called by the dual-window.)

[Examples of usage]

- When the cut and paste positions are specified as follows:

```
    ---Start---   ---End---  ---Paste---
     ( 0, 13)     (118,108)   (  5, 29)

SHOT_INF    Screen;

void main()
{
    Screen.x      = 0;           /* Cut position - Horizontal  */
    Screen.y        = 13;          /* Cut position - Vertical    */
    Screen.x_size   = 118-0+1;          /* Horizontal size      */
    Screen.y_size   = 108-13+1;         /* Vertical size        */
    Screen.p_x      = 5;           /* Paste position - Horizontal */
    Screen.p_y      = 29;          /* Paste position - Vertical   */

    LibScrShot(&Screen);
        •
        •
        •
}
```

- Mode functions -

[Function name] `LibJumpMenu`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibJumpMenu(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Calls the MENU mode.

- Mode functions -

[Function name] `LibGetMode`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGetMode(word *m_code, word *m_sts, word *m_seg, word *m_ofs)
```

[Arguments]

```
word    *m_code    :OUT    Mode code  High-order: Main mode
                                      Low-order: Sub mode
word    *m_sts     :OUT    Mode status
word    *m_seg     :OUT    Segment information
word    *m_ofs     :OUT    Offset information/data pointer
```

[Return values] `None`

[Description] Gets various mode information.

- Mode functions -

[Function name] `LibDualWin`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void far *LibDualWin(word m_code,byte m_sts,void far *ptr);
```

[Arguments]

```
word       m_code   :IN    Mode code   High-order: Main mode
                                       Low-order: Sub mode
byte       m_sts    :IN    Mode status
void far  *ptr      :IN    Data pointer
```

[Return values] `Data pointer`

[Description] Starts up the dual-window and gets the data pointer that has been handled by the dual-processing side.

- Mode functions -

[Function name] `LibDualWinExit`

[Syntax]
```
     #include    "define.h"
     #include    "libc.h"
     bool LibDualWinExit(void far *ptr)
```

[Arguments]
```
     void far    *ptr   :IN     Data pointer
```

[Return values]
```
        Execution result   TRUE: Succeeded
                           FALSE:  Failed
```

[Description] Quits the dual-window, and returns to the place started up.

- Mode functions -

[Function name] `LibModeJump`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibModeJump(word m_code,byte m_sts)
```

[Arguments]

```
word    m_code     :IN   Mode code   High-order: Main mode
                                     Low-order: Sub mode
byte    m_sts      :IN   Mode status
```

[Return values] `Execution result` `TRUE: Succeeded`
`FALSE: Failed`

[Description] Jumps to the mode specified by "m_code".

- Mode functions -

[Function name] `LibScrtJmp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibScrtJmp(byte m_sts,word m_ofs)
```

[Arguments]

```
byte    m_sts       :IN     Mode status
                    0x00: Transfer from normal state by the secret key.
                    IB_MSCRT_MOVE:  Transfer from move selection.
word    m_ofs       :IN     Data pointer
```

[Return values] `None`

[Description] Jumps to the intermediate state for transiting to the Secret mode. (Intermediate state = Password input screen) When specifying IB_MSCRT_MOV to "m_sts", the jump accompanies data transfer.

- Mode functions -

[Function name] `LibSecretCall`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibSecretCall(word m_seg,word m_ofs)
```

[Arguments]

```
word    m_seg      :IN     Segment information
word    m_ofs      :IN     Offset information
```

[Return values] `None`

[Description] Calls the Secret mode by the function specification. The function splits the area allocated as the external variable into "m_seg" and "m_ofs" and transfers them.

[Note] The function to be specified should correspond to the Secret mode.

- Mode functions -

[Function name] `LibScrtModeJmp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibScrtModeJmp(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Jumps from the intermediate state of the Secret mode transition to other mode.

- Mode functions -

[Function name] `LibCrdlOpnJmp`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCrdlOpnJmp(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Changes the mode status to OPEN mode, and performs a forcible mode jump to PC-Link process.

- Mode functions -

[Function name] `LibMenuJump`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibMenuJump(word m_code)
```

[Arguments]

```
word    m_code     :IN   Mode code     High-order: Main mode
                                       Low-order: Sub mode
```

[Return values] `None`

[Description] Jumps from the MENU to other mode.

- Mode functions -

[Function name] `LibGetLastMode`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    void LibGetLastMode(word *m_code, word *m_sts, word *m_seg, word *m_ofs)
```

[Arguments]

```
     word    *m_code    :OUT    Mode code  High-order: Main mode
                                           Low-order: Sub mode
     word    *m_sts     :OUT    Mode status
     word    *m_seg     :OUT    Segment information
     word    *m_ofs     :OUT    Offset information/data pointer
```

[Return values] `None`

[Description] Gets the previous mode information (last time only).

- Mode functions -

[Function name] `LibDataCom`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibDataCom(void);
```

[Arguments] `None`

[Return values] `None`

[Description] Calls the Data Communication process.

- Mode functions -

[Function name] `LibCallListMenu`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCallListMenu(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Calls the list type menu.

- Mode functions -

[Function name] `LibPassWordCheck`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
bool LibPassWordCheck(void)
```

[Arguments] None

[Return values] bool Check result TRUE: Password matched.
FALSE: Password input is aborted by ESC.

[Description] Inputs and checks the system password.

[Examples of usage]
```
p_sel = LibPullDown();       /* Pull-down menu */

switch(p_sel){

    case PDWN_AREA:          /* To_secret_area */
        if(LibPassWordCheck()==TRUE){
   /* To the data transfer process between areas. */
        }
        break;
   •
   •
   •
```

- Mode functions -

[Function name] `LibPassWordEdit`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
void LibPassWordEdit(void)
```

[Arguments] None

[Return values] None

[Description] Corrects the system password.

[Examples of usage]
```
p_sel = LibPullDown();      /* Pull-down menu */

switch(p_sel){

    case PDWN_PASS:         /* Password_edit */
      LibPassWordEdit();  /*Calls the password correction process. */
      break;
        •
        •
        •
```

- Mode functions -

[Function name] `LibMoveArea`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     void LibMoveArea(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Moves between Open area and Secret area.

- Mode functions -

[Function name] `LibModeRestart`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibModeRestart(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Restarts the current mode. (Performs a mode jump with the current mode code.)

[Note] Cannot use this in the Dual-Window state. (NOP.) This function cannot be used when the mode start state has the sub-code. Because the low-order 8-bit of the mode code is fixed to 0x00.

[Examples of usage]

```
if(LibFuncLang()==TRUE){     /*Language setting has been changed!! */
    LibModeRestart();
   }
```

- Mode functions -

[Function name] `LibFileCom`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibFileCom(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Transit to the file transfer mode processing.

- Menu functions -

[Function name] `LibWinIcnMsg`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibWinIcnMsg(byte icn, word msg, byte wtyp)
```

[Arguments]

```
byte      icn      :IN      Icon type
                             ICON_NONE:     None
                             ICON_OK:       OK
                             ICON_BADTZ:    X
                             ICON_BIKKURI:  !
                             ICON_COFFEE:   Coffee
                             ICON_TRASH:    Trash box
                             ICON_SIGN:     Hand
                             ICON_SYNC:     Communicating
word      msg      :IN     Message number
byte      wtyp     :IN     Message display type
```

Value	Status	Button	Position
0x00	Kept opened.	None	Middle
0x10	Kept opened.	None	Bottom
0x01	Closed after 1 sec.	None	Middle
0x11	Closed after 1 sec.	None	Bottom
0x02	Check	OK	Bottom
0x03	Check	ESC	Bottom
0x04	Check	SET	Bottom
0x05	Selection	Yes/No	Bottom
0x06	Selection	OK/ESC	Bottom

[Return values]

```
bool   Touch result   wtyp is 5, 6.  TRUE:  Left button. (YES)
                                     FALSE:  Right button. (NO)
```

[Description] Displays the general-purpose message window.
This function displays a built-in 5-language message or graphic data with specified icon type and performs the several display functions specified by “wtyp”.

[Examples of usage]

```
LibWinIcnMsg(ICON_COFFEE,341,1);
          /* DATA_STORED! (Confirmation display after registration.) */
```

- Menu functions -

[Function name] `LibSelWindow`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
byte LibSelWindow(int x, int y, int xs, byte ln, byte np, SLW_TBL *ktb)
```

[Arguments]

```
int      x        :IN    Window top coordinate - Horizontal
int      y        :IN    Window top coordinate - Vertical
int      xs       :IN    <Unused>
byte     ln       :IN    Number of lists   1-
byte     np       :IN    Default reverse position (0-)
SLW_TBL  *ktb     :IN    Message/return value table
                           ktb[].msg:  Message number
                           ktb[].rtv:  Return values
```

[Return values] `byte` Return value corresponding to the selected position. `ktb[Selected position].rtv`

[Description] Displays a selection window to wait for touching, and returns the selected number.
This function displays a window from the start coordinates specified by x and y, reads the specified number of messages by "ln" from "ktb[]", and then displays them.
The position of the reverse bar immediately after messages are displayed is specified by “np”. If “0xff” is specified, the bar is not displayed for the first time.

[Note] The number of "ktb[]" elements should not exceed "ln".

[Supplement] "xs" was the argument for the window size. However, the current function can calculate the size of the message width internally so that the message with the maximum width can fit to the window. So, this argument is no longer used.

[Examples of usage]

```
byte    mes_ans;
SLW_TBL ktb[3] = {{154,LST_ITEM_PAY},{5,LST_ITEM_EXP},{148,LST_ITEM_DSCRPT}};

mes_ans =   LibSelWindow(0,13,0,3,SYS_AMT_ITEM,ktb);
```

- Menu functions -

[Function name] `LibSelWindow2`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
byte LibSelWindow2(int y, byte ln, SLW_TBL *ktb, word msk)
```

[Arguments]

```
int       y         :IN     Window top coordinate - Vertical
byte      ln        :IN     Number of lists (ecxluding title line)   1 -
SLW_TBL   *ktb      :IN     Message/return value table
                              ktb[].msg:  Message number
                              ktb[].rtv:  Return values
word      msk       :IN     Display item mask information
          When displaying all items: 0x0000
          When not displaying the first line: Turn on a bit of 0x0001.
          When not displaying the second line: Turn on a bit of 0x0002.
          When not displaying the third line: Turn on a bit of 0x0004.
                    •
                    •
                    •
          When not displaying the 15th line: Turn on a bit of 0x8000.
```

[Return values]

```
byte    Return value corresponding to the selected position.
        ktb[Selected position].rtv
```

[Description] Selects a window with the title line. Uses for the deletion menu, etc.
Displays the selection window to wait for touching, and returns the selected number.
Displays a window in the center of the screen from the ordinate specified by "y", reads the specified number of messages by “ln” from “ktb[]”, and then displays them.
ktb[0] is the title line.

[Supplement] When using this function for the deletion menu, set "28" to "y". (Standard specification as of July 2, 1998.)

[Note] Both the title line and the list data are controlled by "ktb[]". However, be sure that the numeric number excluding the title line should be set to "ln" as the number of lists.

[Examples of usage]

```
#define ALL_DATA        0
#define ONE_DATA        1

byte        mes_ans;
SLW_TBL     ktb[3];
```

```
word        msk = 0x0000;

/* Title line */
ktb[0].msg = 23;    /* Delete */

/* Delete one item. */
ktb[1].msg = 57;
ktb[1].rtv = ONE_DATA;

/* Delete all items. */
ktb[2].msg = 56;
ktb[2].rtv = ALL_DATA;

/* If data presents at the cursor position? */
if(AmtLstBuf[AmtLstCurLine].exist_flg==FALSE){
    msk = 0x0001;   /* Mask one item data. */
}
mes_ans = LibSelWindow2(12,2,ktb,msk);
```

- Menu functions -

[Function name] `LibSelWindowExt`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
byte LibSelWindowExt(SLW2_ST *slw)
```

[Arguments]

```
SLW2_ST     *slw    : IN     Selection window information
```

[Return values]

```
Return value corresponding to the selected position.
Table elements specified by "slw->rtv".
```

[Description] Performs the selection window processing. This is a character string transfer version of LibSelWindow().
This function displays the selection window to wait for touching, and returns the selected number.
The function displays a window with a size specified by “xs” from the start coordinates specified by x and y, reads messages from "slw->cmnt”, and displays them.
The position of the reverse bar immediately after messages are displayed is specified by "np”. If "0xff" is specified, the bar is not displayed for the first time.

[Supplement] Selection window information

slw->x	:	Start coordinate of the window. Horizontal
slw->y	:	Start coordinate of the window. Vertical
slw->xs	:	Horizontal size of the window.
slw->ity	:	Line spacing of message list (9 -)
slw->np	:	Default position of the highlighted cursor (no first time display with "0xff").
slw->cmnt	:	Display message buffer (separated by "0xfe", ended by "0xff").
slw->rtv	:	Return value relevant to the selected message.
slw->t_xs	:	Message display start position in the window.

[Examples of usage]

```
byte     rtv[10+1] = {0,1,2,3,4,5,6,7,8,9,0xff};
byte     cmnt[10*(14+1)],np;
SLW2_ST win_prm;

byte     sel_ret;
byte     t_tbl[10][14+1];
int       i,k,len;

/* Source data*/
strcpy(t_tbl[0],"Mileage");
strcpy(t_tbl[1],"Fuel");
strcpy(t_tbl[2],"Parking&Tolls");
strcpy(t_tbl[3],"Taxi");
strcpy(t_tbl[4],"Meals");
strcpy(t_tbl[5],"Phone");
strcpy(t_tbl[6],"Entertainment");
strcpy(t_tbl[7],"Hotel");
strcpy(t_tbl[8],"Miscellaneous");
strcpy(t_tbl[9],"FREE");

/* Gets an initial value. */
np = 0;

/* Creates a display buffer. */
for(i=k=0;i<10;i++,k++){
    len = strlen(t_tbl[i]);
    memcpy(&cmnt[k],t_tbl[i],len);
    k += len;
    if(i<9) cmnt[k] = 0xfe;     /* Data separation */
    else    cmnt[k] = 0x00;     /* Final data.   */

}

/* The selection window processing */
win_prm.x    =   9;
win_prm.y    =  32;
win_prm.xs   = 119;
win_prm.ity  =   9;
win_prm.np   = np;
win_prm.cmnt = cmnt;
win_prm.rtv  = rtv;
win_prm.t_xs =   3;
sel_ret = LibSelWindowExt(&win_prm);
```

- Menu functions -

[Function name] `LibSelWinExt2A`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
byte LibSelWinExt2A(SLW2_ST *slw)
```

[Arguments]

```
SLW2_ST     *slw :IN     Selection window information
```

[Return values] `byte` Number of lists in the window.

[Description] Displays a window by the message list. (The character string transfer.)
Displays the message list from the contents specified by "slw[]"
This function does not include the touch waiting as it is different from LibSelWindowExt() and so on. Therefore, after performing the message list display by this function, any effects can be made in the window.

[Note] No touch waiting is supported. So, this function needs to be paired with LibSelWinExt2B() when using.

- Menu functions -

[Function name] `LibSelWinExt2B`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
byte LibSelWinExt2B(SLW2_ST *slw, byte ln)
```

[Arguments]

```
SLW2_ST   *slw      : IN     Selection window information
byte      ln        : IN     Number of lists (1-)
```

[Return values]

```
byte  Return value corresponding to the selected position.
      ktb[Selected position].rtv
```

[Description] Waits for touching of the message list based on the contents specified by "slw[]".

[Note] No display processing is supported. So, this function needs to be paired with LibSelWinExt2A() when using.

- Menu functions -

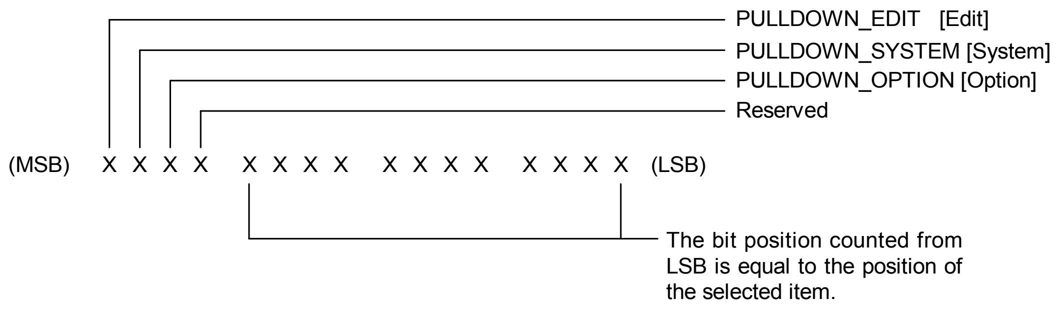
[Function name] `LibPullDown`

[Syntax]

```
#include    "define.h"
#include    "PullDown.h"
word LibPullDown(void)
```

[Arguments] `None`

[Return values] `Selected items.`


PULLDOWN_EDIT [Edit]
PULLDOWN_SYSTEM [System]
PULLDOWN_OPTION [Option]
Reserved
(MSB) X X X X X X X X X X X X X X X X (LSB)
The bit position counted from LSB is equal to the position of the selected item.

[Description] Displays a pull-down menu and returns a selected item.
It is absolutely necessary to call LibPullDownInit().

[Note] Among the messages given to the SYSTEM columns, "36th" is for the process name of the language setting function. Depending on the ROM models, there is a specification that suppresses the selection of this process. In that case, this function forcibly masks it. Therefore, the "36th" can not be used for other purposes than the process name of the language setting function.
The ROM model can be checked internally by LibGetLangInf().

[Examples of usage]

```
LibPullDownInit( IdxEdt,IdxSys,IdxOpt ); /* Initializes the pull-down.*/
sel = LibPullDown();
if( sel& PULLDOWN_OPTION){
   sel &= ~PULLDOWN_IDX_MASK;  /*Bit for item.*/
}
```

- Menu functions -

[Function name] `LibPullDownInit`

[Syntax]
```
     #include    "define.h"
     #include    "PullDown.h"
     int LibPullDownInit(word *edt, word *sys, word *opt)
```

[Arguments]
```
     word *edt          :IN [Edit] item message code array
     word *sys          :IN [System] item message code array
     word *opt          :IN [Option] item message code array
```

[Return values] `Normal "0" / Abnormal "-1"`

[Description] Initializes the pull-down menu display.
It is absolutely necessary to call up LibPullDown().
The end of array must be put since it is recognized by PDNTBLEND.
The item is allocated to bits from the lowest position during registration.
(See LibPullDown())

[Examples of usage]
```
     word IdxEdt[] = {
        20,    /* CUT */
        21,    /* COPY       */
         •
         •
         •
        PDNTBLEND
     };

     word IdxSys[] = {
        38,    /* SET DATE/TIME     */
        33,    /* SOUND       */
         •
         •
         •
        PDNTBLEND
     };

     word IdxOpt[] = {
        32,        /* FULL_SCREEN */
        PDNTBLEND
     };
     word sel;

     LibPullDownInit( IdxEdt,IdxSys,IdxOpt ); /* Initializes the pull-down*/
```

```
sel = LibPullDown();
```

- Menu functions -

[Function name] `LibPullDownAtrSet`

[Syntax]

```
#include    "define.h"
#include    "PullDown.h"
int LibPullDownAtrSet(int mode,word type,word item)
```

[Arguments]

```
int   mode     :IN Attribute type
               PULLDOWN_NONDSP   No display
               PULLDOWN_HTNDSP   Dimmed (grayed) display (Reserved)

word  type     :IN Heading type
               PULLDOWN_EDIT [Edit]
               PULLDOWN_SYSTEM   [System]
               PULLDOWN_OPTION   [Option]

word item      :IN Specifies destination item (supports bits from LSB).
```

[Return values] `Normal "0" / Abnormal "-1"`

[Description] Sets the attribute of the pull-down menu display.

[Examples of usage]

```
word IdxEdt[] = {
   20,    /* CUT */
   21,    /* COPY       */
    •
    •
    •
   PDNTBLEND
};

word IdxSys[] = {
   38,    /* SET DATE/TIME     */
   33,    /* SOUND       */
    •
    •
    •
   PDNTBLEND
};

word IdxOpt[] = {
   32,        /* FULL_SCREEN */
   PDNTBLEND
};
word sel;


    •
    •
    •

LibPullDownInit( IdxEdt,IdxSys,IdxOpt ); /* Initializes the pull-down*/

/* [Edit] Specifies item to be shaded.*/
LibPullDownAtrSet( PULLDOWN_HTNDSP, PULLDOWN_EDIT  ,
     PULLDOWN_EDT_CUT
    |PULLDOWN_EDT_SEARCH
);

/*[System] Specifies item to be shaded. */
LibPullDownAtrSet( PULLDOWN_HTNDSP, PULLDOWN_SYSTEM,
     PULLDOWN_SYS_SET_DATE_TIME
    |PULLDOWN_SYS_DATA_COMMUNICATION
);

/* [Option] Specifies item to be shaded. */
LibPullDownAtrSet( PULLDOWN_HTNDSP, PULLDOWN_OPTION,
     PULLDOWN_SYS_MULTIPLE_HIGHLIGHT
);

sel = LibPullDown();
```

- Menu functions -

[Function name] `LibEditPullDown`

[Syntax]

```
#include    "define.h"
#include    "PullDown.h"
void LibEditPullDown(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Performs the pull-down menu process during item input.
Performs the pattern selection in order to select only 4 items of the SYSTEM.

```
 EDIT                  SYSTEM                     OPTION
---------------------------------------------------------
                       Sound
                       Capacity
                       Contrast
                       Touch_Panel_Alignment
```

- Menu functions -

[Function name] `LibSelWinLckA`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibSelWinLckA(int x, int y, int ity, int xs, byte ln, SLW_TBL *ktb)
```

[Arguments]

```
int       x         :IN     Window display start coordinate (X)
int       y         :IN     Window display start coordinate (Y)
int       ity       :IN     Display line spacing dot numbers (9 or 10)
int       xs        :IN     Window display width (X size)
byte      ln        :IN   Display item line numbers in window (1, 2, 3, ...).
SLW_TBL   *ktb      :IN     Message/return value table
                              ktb[].msg:  Message number
                              ktb[].rtv:  Return values
```

[Return values] `None`

[Description] Processes the fixed message window display.

[Note] No display process is supported. So, this function needs to be paired with LibSelWinLckB() when using.

- Menu functions -

[Function name] `LibSelWinLckB`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
byte LibSelWinLckB(int x, int y, int ity, int xs, byte ln, byte np, SLW_TBL *ktb)
```

[Arguments]

```
int       x        :IN   Window display start coordinate (X)
int       y        :IN   Window display start coordinate (Y)
int       ity      :IN   Display line spacing dot numbers (9 or 10)
int       xs       :IN   Window display width (X size)
byte      ln       :IN   Display item line numbers in window (1, 2, 3, ...).
byte      np       :IN   Reversed cursor default position in window (0, 1, 2, ...).
                                      (No default reverse with 0xff.)
SLW_TBL   *ktb     :IN   Message/return value table
                         ktb[].msg:  Message number
                         ktb[].rtv:  Return values
```

[Return values]

```
byte  Return value corresponding to the selected position
      ktb[Selected position].rtv
```

[Description] Waits for touching of the message list specified by "ktb[]".

[Note] No display process is supported. So, this function needs to be paired with LibSelWinLckA() when using.

[Examples of usage]

```
SLW_TBL ktb[3] = {{259,0xa0},{260,0xa1},{148,0xa2}};
byte    mes_ans;
int     x,y,ity,xs;
byte    ln;

x   = 13;
y   = 12;
ity = 9;
xs  = 100;
ln  = 3;

LibSelWinLckA(x,y,ity,xs,ln,ktb);
mes_ans = LibSelWinLckB(x,y,ity,xs,ln,0,ktb);
```

- Menu functions -

[Function name] `LibSelectFont`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibSelectFont(byte far *font)
```

[Arguments]

```
byte far  *font     :IN/OUT   Font information   IB_PFONT1
                                                 IB_PFONT2
```

[Return values]

```
bool   Changed/Not Changed    TRUE:  Changed.
                              FALSE:  Not changed.
```

[Description] Displays the selection window to change the display font. Selects a font from two font types in the window, and the function returns TRUE if the selection is changed.

[Examples of usage]

```
bool    sel_font;

sel_font = LibSelectFont(&SYS_EXP_FONT);

if(sel_font == TRUE){
    /* To the data re-display process. */
}
```

- Menu functions -

[Function name] `LibSelWinOpen2`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibSelWinOpen2(int x, int y, byte iy, int xs,
                    byte ln, SLW_TBL *ktb, word msk)
```

[Arguments]

```
int     x          : Starting coordinate to display window (X)
int     y          : Starting coordinate to display window (Y)
byte    iy         : Space between displayed items (Y)
int     xs         : Displayed width of window (X)
byte    ln         : Lines in selection window
SLW_TBL *ktb       : Struct pointer for selection window
word    msk        : Displayed item masking information
```

[Return values] None

[Description] Open the selection window and display comments in the window.

- Menu functions -

[Function name] `LibSelectWin`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
byte LibSelectWin(int x, int y, byte iy, int xs,  byte ln, byte np)
```

[Arguments]

```
int   x  :IN Starting coordinate to display window (X)
int   y  :IN Starting coordinate to display window (Y)
byte  iy :IN Space between displayed items (Y)
int   xs :IN Displayed width of window ((X)
byte  ln :IN Lines in selection window
byte  np :IN Position of default inversion (0xff when no default inversion)
```

[Return values] None

[Description] Wait for touch and inverse the cursor.

- Menu functions -

[Function name] `LibSelWinTchSet`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibSelWinTchSet(void)
```

[Arguments] None

[Return values] `None`

[Description] Perform touch area definitions in the selection window.
(Specifically, full screen area, action keys and hardware icons)

- System functions -

[Function name] `LibSaveSysRam`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSaveSysRam(void)
```

[Arguments] `None`

[Return values] `FLASH BIOS status code.`

[Description] Saves all system area data for application to the FLASH memory.

- System functions -

[Function name] `LibSaveSysRamB`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSaveSysRamB(void)
```

[Arguments] `None`

[Return values] `FLASH BIOS status code.`

[Description] Saves all system area data for BIOS to the FLASH memory.

- System functions -

[Function name] `LibGetBLD`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     bool LibGetBLD(void)
```

[Arguments] `None`

[Return values]	`Inspection result`	`TRUE: Normal`
		`FALSE: Low battery.`

[Description] Checks the battery status.

- System functions -

[Function name] `LibGetVersion`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibGetVersion(byte *ver_str)
```

[Arguments]

```
byte    *ver_str    :OUT    Version string    [12+1]
```

[Return values] None

[Description] Gets the ROM version.
Outputs the ROM creation date/time to the buffer specified by "ver_str" in ASCII format string.

[Examples of usage]

```
byte    ver_str[12+1];

LibGetVersion(ver_str);     /* "199804241536" ->1998/4/24 15:36 */
```

- System functions -

[Function name] `LibELHandle`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    #include    "l_define.h"
    #include    "l_libc.h"
    void LibELHandle(byte mode)
```

[Arguments]

```
    byte      mode     :IN   ON type  IB_ELP_OFF: Turns OFF.
                                      IB_ELP_ON: Turns ON.
                                      IB_ELP_SON: Lights continuously.
```

[Return values] `None`

[Description] Performs various EL-panel operations.

- System functions -

[Function name] `LibGetEL`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
byte LibGetEL(void)
```

[Arguments] `None`

[Return values]

```
byte    State    IB_ELP_OFF: Turns OFF.
                 IB_ELP_ON:  Turns ON.
                 IB_ELP_SON: Lights continuously.
```

[Description] Gets the EL-panel status.

- System functions -

[Function name] `LibGetLang`

[Syntax]
```
     #include    "define.h"
     #include    "libc.h"
     #include    "l_define.h"
     #include    "l_libc.h"
     byte LibGetLang(void)
```

[Arguments] `None`

[Return values]

byte	Language information	IB_DEUTSCH:	German
		IB_ENGLISH:	English
		IB_ESPANOL:	Spanish
		IB_FRANCAIS:	French
		IB_ITALIANO:	Italian

[Description] Gets the current language information of the system.

- System functions -

[Function name] `LibSetLang`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibSetLang(byte lang)
```

[Arguments]

```
byte    lang        :IN     Language information
                                 IB_DEUTSCH: German
                                 IB_ENGLISH: English
                                 IB_ESPANOL: Spanish
                                 IB_FRANCAIS: French
                                 IB_ITALIANO: Italian
```

[Return values] `None`

[Description] Sets and changes the system language information.

[Supplement] When the language information is changed, at the same time the keyboard layout information is also updated. The keyboard layout information can also be changed using LibSetKeyKind(). However, when the language is changed, the appropriate initial value is used instead.

```
English:   QWERTY layout
French:    AZERTY layout
German:    QWERTZ layout
Italian:   QWERTY layout
Spanish:   QWERTY layout
```

- System functions -

[Function name] `LibSoundGet`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
byte LibSoundGet(void)
```

[Arguments] `None`

[Return values]

```
byte  Sound information (various ON information)
                    IX_DAYLY_ALM: Daily alarm
                    IX_DATA_ALM: Data alarm
                    IX_KEY_SOUND: Key sound
```

[Description] Gets the current sound information of the system.

- System functions -

[Function name] `LibSoundSet`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibSoundSet(byte opt)
```

[Arguments]

```
byte    opt        :IN     Sound information (bit specification)
                             IX_DAYLY_ALM: Daily alarm
                             IX_DATA_ALM: Data alarm
                             IX_KEY_SOUND: Key sound
```

[Return values] `None`

[Description] Sets and changes the system sound information.

- System functions -

[Function name] `LibContrastInit`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     #include    "l_define.h"
     #include    "l_libc.h"
     bool LibContrastInit(void)
```

[Arguments] `None`

```
[Return values]      bool   Execution result    TRUE: Succeeded
                                                FALSE: Failed
```

[Description] Initializes the contrast setting. (Restores the factory defaults.)

- System functions -

[Function name] `LibContrastUp`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     #include    "l_define.h"
     #include    "l_libc.h"
     bool LibContrastUp(void)
```

[Arguments] `None`

[Return values]

```
bool   Execution result   TRUE: Succeeded
                          FALSE: Failed (No more darkness.)
```

[Description] Adjusts the contrast setting one level darker.

- System functions -

[Function name] `LibContrastDown`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
bool LibContrastDown(void)
```

[Arguments] `None`

[Return values]

```
bool  Execution result  TRUE: Succeeded
                        FALSE: Failed (No more lightness.)
```

[Description] Adjusts the contrast setting one level lighter.

- System functions -

[Function name] `LibDigitizer`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
byte LibDigitizer(word *obj)
```

[Arguments]

```
word     *obj     :OUT    Hardware icon object code
                           (For startup from OFF.)
```

[Return values]

```
byte  End status  IB_NOERR_END: Normal end
                  IB_ESC_END: Ends by ESC icon or after OFF key.
                  IB_ALM_END: Ends by alarm matching.
                  IB_PON_END: Starts up from off mode.
```

[Description] Adjusts the touch-panel.

- System functions -

[Function name] `LibPassClr`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    #include    "l_define.h"
    #include    "l_libc.h"
    void LibPassClr(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Clears the system password.

- System functions -

[Function name] `LibPassSet`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    #include    "l_define.h"
    #include    "l_libc.h"
    void LibPassSet(byte far *pas_buf)
```

[Arguments]

```
    byte far  *pas_buf           :IN     Password string
```

[Return values] `None`

[Description] Sets and changes the system password.

[Note] Puts a NULL(0x00) to the end of "pas_buf[]". The maximum number of characters is 16 characters.

- System functions -

[Function name] `LibPassGet`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    #include    "l_define.h"
    #include    "l_libc.h"
    int LibPassGet(byte far *pass_buf)
```

[Arguments]

```
    byte far  *pass_buf         :OUT    Output destination password string
```

[Return values] `Password string length`

[Description] Gets the system password. Outputs the password string to the string buffer specified by "pass_buf".

- System functions -

[Function name] `LibPassChk`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    #include    "l_define.h"
    #include    "l_libc.h"
    bool LibPassChk(byte far *pass_buf)
```

[Arguments]

```
    byte far  *pass_buf        :IN     Password string
```

[Return values]

```
bool    Inspection result       TRUE:  Password matched.
                                FALSE: Password unmatched.
```

[Description] Checks the password.
Checks whether the password string specified by "pass_buf" is equal to the registered password.

- System functions -

[Function name] `LibGetAPOTime`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
word LibGetAPOTime(void)
```

[Arguments] `None`

[Return values] `word     APO time (500-ms units)`

[Description] Gets the APO time.

- System functions -

[Function name] `LibSetAPOTime`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     #include    "l_define.h"
     #include    "l_libc.h"
     void LibSetAPOTime(word msec)
```

[Arguments]

```
     word    msec       :IN     APO time (500-ms units)
```

[Return values] `None`

[Description] Sets the APO time. Automatically sets the APO time to 6 minutes if attempting to set an illegal value.

- System functions -

[Function name] `LibSetKeyKind`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibSetKeyKind(byte type)
```

[Arguments]

```
byte    type       :IN     Keyboard layout type
                             IB_QWERTY:  QWERTY layout
                             IB_AZERTY:  AZERTY layout
                             IB_QWERTZ:  QWERTZ layout
```

[Return values] `None`

[Description] Sets the keyboard layout type.

[Note] The keyboard layout information can be set/changed not only by this function but also by the language setting LibSetLang(). When the language setting LibSetLang() is executed, the keyboard layout information is automatically changed to the appropriate initial value.

```
English:   QWERTY layout
French:    AZERTY layout
German:    QWERTZ layout
Italian:   QWERTY layout
Spanish:   QWERTY layout
```

- System functions -

[Function name] `LibGetKeyKind`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
byte LibGetKeyKind(void)
```

[Arguments] `None`

[Return values]

```
Keyboard-layout type  IB_QWERTY: QWERTY layout
                      IB_AZERTY: AZERTY layout
                      IB_QWERTZ: QWERTZ layout
```

[Description] Gets the keyboard layout type.

- System functions -

[Function name] `LibBuzzerOff`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    #include    "l_define.h"
    #include    "l_libc.h"
    void LibBuzzerOff(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Turns off a buzzer.

- System functions -

[Function name] `LibBuzzerOn`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
#include   "l_define.h"
#include   "l_libc.h"
void LibBuzzerOn(byte b_type)
```

[Arguments]

```
byte    b_type     :IN      Buzzer type
                             IB_BEEP0_SET: Through (Sounds for 1-sec.)
                             IB_BEEP1_SET: 1 time per second
                             IB_BEEP2_SET: 2 times per second
                             IB_BEEP3_SET: 3 times per second
```

[Return values] None

[Description] Sounds a buzzer.

- System functions -

[Function name] `LibGetLangInf`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
#include   "l_define.h"
#include   "l_libc.h"
byte LibGetLangInf(void)
```

[Arguments] `None`

[Return values]

```
byte  Language support information
     IB_LANG_ENGLISH:  A single language version. (English only)
              Others:  5-language version
```

[Description] Gets the information whether the current ROM model is the single language version or the 5-language version.

- System functions -

[Function name] `LibInitial`

[Syntax]
```
    #include   "define.h"
    #include   "libc.h"
    bool LibInitial(bool ClrFlg)
```

[Arguments]
```
    bool    ClrFlg     :IN      Indicate clear
                                :TRUE     Indicate full initialization
                                :FALSE    Initialization from normal ON only
```

[Return values] bool ret

:TRUE	Normal
:FALSE	Abnormal

[Description] Perform the following initialization items (full initialization):

- Set initial value of Csysram area
- Set initial value of library global area
- Reserve save area of screen for BLD and alarm
  (Bdisp_Save_SaveDisp)
  Allocate the following memory areas:
  - Work area for flash write: 32KB plus
  - Add-in program data information: 8KB
  - Save area for Window ID: 0.5KB
- Set date (set 2003.01.01)
  (Btime_GetTime_Rtc)
- Reserve library global area
  (Bmem_Allocate Reserve by system specification)
- Register alarm driver
  (Balarm_RegisterAlarmDriver)
- Register daily alarm format
  (Balarm_RegisterDailyAlarm)
- Set language (temporary execution)
  (LibB_SetLang_Message)

According to the parameter, initialization from normal ON does not perform all the full initialization items.
It currently performs only:
- Clear of flag whose password has been set (CSYSRAM).

- System functions -

[Function name] `LibInitial2`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
bool LibInitial2(void)
```

[Arguments] `None`

[Return values]

```
bool    ret          :TRUE      Normal
                     :FALSE     Abnorma
```

[Description] Perform library initialization (initialization regarding flash access).
Though there are currently no initialization items, items will be added as necessary.

- System functions -

[Function name] `LibGetCommDevice`

[Syntax]

```
    #include   "define.h"
    #include   "libc.h"
    byte LibGetCommDevice(void)
```

[Arguments] `None`

[Return values]

```
byte  Selected communication device
  IB_SRL_COM2: Serial 9 pin selected
  IB_SRL_COM3: USB
```

[Description] Get the currently selected communication device.

- System functions -

[Function name] `SysGetPONstat`

[Syntax]

```
     #include   "define.h"
     #include   "libc.h"
     bool SysGetPONstat(void)
```

[Arguments] `None`

[Return values]

```
bool   TRUE: Enable power on by pressing touch panel
       FALSE: Disable power on by pressing touch panel.
```

[Description] Get Touch PowerON setting information.

- System functions -

[Function name] `SysSetPONstat`

[Syntax]

```
     #include   "define.h"
     #include   "libc.h"
     void SysSetPONstat(bool OnStat)
```

[Arguments]

```
     bool  OnStat       TRUE : Opening specification screen display on
                        FALSE: Opening specification screen display off
```

[Return values] `None`

[Description] Set Touch PowerON setting information.

- System functions -

[Function name] `SysGetSUPstat`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
bool SysGetSUPstat(void)
```

[Arguments] `None`

[Return values]

```
bool   TRUE: Opening specification screen display on
       FALSE: Opening specification screen display
```

[Description] Get on/off state of opening specification screen display.

- System functions -

[Function name] `SysSetSUPstat`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
void SysSetSUPstat(bool OnStat)
```

[Arguments]

```
bool OnStat        TRUE : Enable power on by pressing touch panel
                   FALSE: Disable power on by pressing touch panel
```

[Return values] `None`

[Description] Set opening specification screen display on/off.

- System functions -

[Function name] `SysGetELTime`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
byte SysGetELTime(void)
```

[Arguments] `None`

[Return values]

```
byte elt     0: 15 sec
             1: 30 sec
             2: 60 sec
```

[Description] Get the length of EL backlight (60, 30 or 15 seconds).

- System functions -

[Function name] `SysSetELTime`

[Syntax]

```
#include   "define.h"
#include   "libc.h"
void SysSetELTime(byte elt)
```

[Arguments]

```
byte elt          0: 15 sec
                  1: 30 sec
                  2: 60 sec
```

[Return values] `None`

[Description] Set the length of EL backlight (60, 30 or 15 seconds).

- Function functions -

[Function name] `LibFuncDateTime`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
void LibFuncDateTime(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Calls the date/time setting process.

- Function functions -

[Function name] `LibFuncSound`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    void LibFuncSound(bool IsAlarmOnly)
```

[Arguments]

```
    bool    IsAlarmOnly             :IN    Setting limitation
                               Except "0": Only alarm setting.
                               "0": Possible to change all.
```

[Return values] `None`

[Description] Calls the sound information setting process.

- Function functions -

[Function name] `LibFuncFormat`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibFuncFormat(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Calls the setting process of the various FORMATS.

- Function functions -

[Function name] `LibFuncLang`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibFuncLang(void)
```

[Arguments] `None`

[Return values]

`bool`	`Changed/Not changed`	`TRUE: Changed.`
		`FALSE: Not changed.`

[Description] Calls the language change process.
If the language currently used is changed, the function returns TRUE.

[Examples of usage]

```
if(LibFuncLang()==TRUE){
    LibModeJump(IW_MEXPEN | IB_SMALL,0x00); /* Restarts mode. */
}
```

- Function functions -

[Function name] `LibFuncCapa`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibFuncCapa(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Calls the FLASH memory capacity displays process.

- Function functions -

[Function name] `LibFuncContrast`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibFuncContrast(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Calls the CONTRAST setting process.

- Function functions -

[Function name] `LibFuncDigitizer`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibFuncDigitizer(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Calls the touch-panel adjustment process.

- Function functions -

[Function name] `LibFuncMemoryManagement`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibFuncMemoryManagement(void)
```

[Arguments] `None`

[Return values] `None`

[Description] Calls the memory management process.

- Function functions -

[Function name] `LibFuncPtool`

[Syntax]

```
     #include    "define.h"
     #include    "libc.h"
     bool LibFuncPtool(void)
```

[Arguments] `None`

[Return values]

`bool`	`Date change/not changed`	`TRUE: Changed.`
		`FALSE: Not changed.`

[Description] Calls the pop-up tool.

[Note] This function executes nothing when the mode status works in the Dual-Window.
As of June 1, 1999: The return value always returns FALSE unconditionally.
(Will be maintained.)

- Function functions -

[Function name] `LibCalWin`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCalWin(byte tch_btn)
```

[Arguments]

```
byte    tch_btn  :IN    Loading function of arithmetic operation result.
                    ON:     Loading
                    OFF:    No loading
```

[Return values] `None`

[Description] Calls the Calculator of the pop-up tool.
If "tch_btn" is ON, a button is provided, which is used to write calculation results to the copy buffer.

- Function functions -

[Function name] `LibFuncUSB`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibFuncUSB(void)
```

[Arguments] None

[Return values]

```
bool   Changed/unchanged  TRUE: Changed
                          FALSE: Unchanged
```

[Description] Call USB selection.
This returns TRUE if the current USB selection is changed..

- Calculator functions -

[Function name] `LibCalBase`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCalBase(CALWRAM *calram,byte kind)
```

[Arguments]

```
CALWRAM   *calram   :IN/OUT   Calculator data buffer
byte       kind     :IN       Arithmetic operation   0x00: +
                                                     0x01: -
                                                     0x02: ×
                                                     0x03: ÷
```

[Return values] `None`

[Description] Performs Calculator's four basic calculations.

- Calculator functions -

[Function name] `LibCalBase2`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    bool LibCalBase2(byte *a_dat,const byte *n_dat1,
                              const byte *n_dat2,byte kind)
```

[Arguments]

```
    byte *a_dat  :OUT  Calculation result buffer [CAL_EZSIZE]
    const byte  *n_dat1         :IN  Operation object buffer 1 [CAL_EZSIZE]
    const byte  *n_dat2         :IN  Operation object buffer 2 [CAL_EZSIZE]
    byte     kind               :IN  Kind of operation  0x00: +
                                                        0x01: -
                                                        0x02: A~
                                                        0x03: AA
                                                        0x0b: A,,
```

[Return values]

```
    bool  TRUE: Operation normal
          FALSE: Operation abnormal
```

[Description] Perform the four basic and square root operations with scientific notation buffers.
*n_dat1 (+,-,A~,AA,A,,) *n_dat2.

- Calculator functions -

[Function name] `LibCalBaseData`

[Syntax]

```
  #include    "define.h"
  #include    "libc.h"
  void LibCalBaseData(byte *a_dat,const byte *n_dat1,const byte *n_dat2,byte
kind)
```

[Arguments]

```
     byte        *a_dat    :OUT    Calculation result buffer    [CAL_EZSIZE]
     const byte  *n_dat1   :IN     Operation destination buffer1  [CAL_EZSIZE]
     const byte  *n_dat2   :IN     Operation destination buffer2  [CAL_EZSIZE]
     byte        kind      :IN     Operation types  0x00: +
                                                    0x01: -
                                                    0x02: ×
                                                    0x03: ÷
```

[Return values] None

[Description] Performs Calculator's four basic calculations using the operation buffer in the exponential format.
`*n_dat1  kind(+, −, ×, ÷)  *n_dat2`

[Note] Not detecting the operation result error.

[Examples of usage]

```
    byte   dat1[] = {0x10,0x89,0x99,0x99,0x99,0x99,0x99,0x00,0x00,0x00,0x00},
           dat2[] = {0x10,0x61,0x23,0x45,0x67,0x89,0x00,0x00,0x00,0x00,0x00},
             a_dat[CAL_EZSIZE];

/*Outputs the operation result of "999999999.99 - 1234567.89" to "a_dat".*/
    LibCalBaseData(a_dat,dat1,dat2,0x01);
```

- Calculator functions -

[Function name] `LibCalRoot`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCalRoot(CALWRAM *calram)
```

[Arguments]

```
CALWRAM     *calram  :IN/OUT   Calculator data buffer
```

[Return values] `None`

[Description] Performs Calculator's root calculation.

- Calculator functions -

[Function name] `LibCalKeyInit`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCalKeyInit(CALWRAM *calram)
```

[Arguments]

```
CALWRAM     *calram  : OUT    Calculator data buffer
```

[Return values] `None`

[Description] Initializes the Calculator keyboard.
The "calram" operation buffer is cleared to "0". (This is equivalent to the AC key operation.)

- Calculator functions -

[Function name] `LibCalKeyDsp`

[Syntax]
```
     #include    "define.h"
     #include    "libc.h"
     void LibCalKeyDsp(CALWRAM *calram)
```

[Arguments]
```
     CALWRAM     *calram  :IN     Calculator data buffer
```

[Return values] `None`

[Description] Displays the Calculator keyboard.

[Note] The "calram" must be initialized using LibCalKeyInit() before executing this function.
This function does not perform data transfer to the D/D. Therefore, newly set data is not displayed actually (invalid) unless LibPutDisp is executed.

- Calculator functions -

[Function name] `LibCalKeyTchWait`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCalKeyTchWait(CALWRAM *calram,TCHSTS *tsts)
```

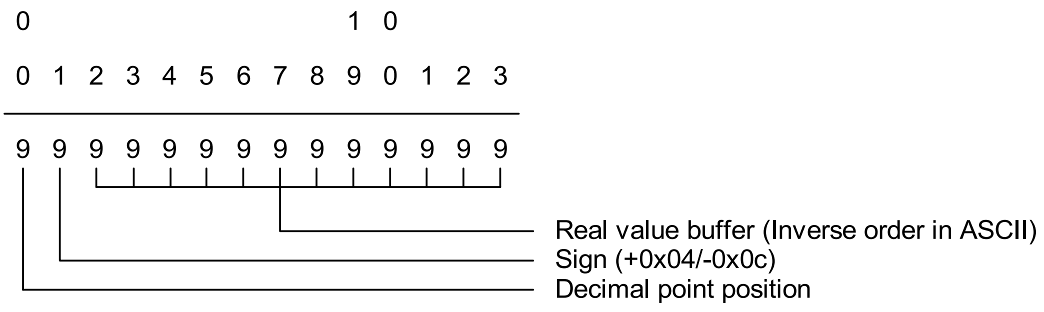
[Arguments]

```
CALWRAM   *calram   :IN/OUT Calculator data buffer
TCHSTS    *tsts     :IN/OUT Touch status information
```

[Return values] None

[Description] Waits for touching of the Calculator keyboard, and performs from the operation processing to the display processing.
If the touch occurs in the area outside the Calculator keyboard, the object code is output to "tsts->obj".
When "NEXT" key or "=" on the Calculator keyboard is pressed, OBJ_CAL_NEXT or OBJ_CAL_EQUAL is output to "tsts->obj". When other Calculator keys are pressed, the operation processing is continued without stepping out from this function.
When loading the calculation result, it is necessary to edit the contents of "calram->calxbuf[]" according to the specification.

[Supplement] "calram->calxbuf[]" format is as follow:



[Example]

```
[  0.]
→{0x0B,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
[           3.14]
→{0x09,0x04,0x34,0x31,0x33,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
[-123456789012.]
→{0x0B,0x0C,0x32,0x31,0x30,0x39,0x38,0x37,0x36,0x35,0x34,0x33,0x32,0x31}
[ 1.41421356237]
→{0x00,0x04,0x37,0x33,0x32,0x36,0x35,0x33,0x31,0x32,0x34,0x31,0x34,0x31}
```

[Examples of usage]
```
#define OBJ_SET         0xe011      /* SET key */
#define OBJ_CLR         0xe012      /* CLR key */

static TCHTBL far TchClr[] =        /* CLR touch information. */
{
    14,0,43,11,ACT_ICON,OBJ_CLR,0x0000,
    0,0,0,0,ACT_NONE,OBJ_END,0x0000
};
static TCHTBL far TchSet[] =        /* SET touch information. */
{
     99,0,128,11,ACT_ICON,OBJ_SET,0x0000,
    0,0,0,0,ACT_NONE,OBJ_END,0x0000
};
static T_ICON far PassIcon[] =      /* Icon data */
{
    TchSet, NULL, NULL,0x02,
    TchClr, NULL, NULL,0x02,
};

void main(void){

    TCHSTS      tsts;
    CALWRAM     calram;

    /*****************/
    /*  Touch setting*/
    /*****************/
    LibTchStackClr();
    LibTchStackPush(TchHardIcon);
    LibTchStackPush(TchModeIcon);
    LibTchStackPush(TchSet);
    LibTchStackPush(TchClr);
    LibTchInit();

    /**********************/
    /* Drawing background */
    /**********************/
    LibClrDisp();
    LibPutFarData( 14,  0,132);         /* Icon display for Clr */
    LibPutFarData( 99,  0,132);         /* Icon display for Set  */
    LibPutMessageCenter(110, 15, 41,2,IB_PFONT1); /*Clr character display */
    LibPutMessageCenter( 60,100,126,2,IB_PFONT1); /*Set character display */
    /****************************/
    /* Initializing calculator. */
    /****************************/
    LibCalKeyInit(&calram);     /* Initializes calculator keyboard. */
    LibCalKeyDsp(&calram);      /* Displays Calculator keyboard.   */

    LibPutDisp();   /* Reflects to the screen. */

    while(1){

        LibCalKeyTchWait(&calram,&tsts); /*Waits for touch of calculator*/

        switch(tsts.obj){
   case OBJ_SET:       /* Set */
       if (LibIconClick(&PassIcon[0], &tsts) == TRUE){
           if(calram.calerror == 0x00){    /* NOR operation error? */
          /***** Writing process and others... *****/
           }
       }
       break;

   case OBJ_CLR:       /* Clr */
       if (LibIconClick(&PassIcon[1], &tsts) == TRUE){
           /***** Clear process and others... *****/
       }
       break;

   case OBJ_CAL_NEXT:  /* "Next" */
       /***** Writing and item move processes, etc. *****/
       break;

   case OBJ_CAL_EQUAL: /* "="   */
       /***** Writing process and others... *****/
       break;

   case OBJ_HIC_ESC:   /* ESC   */
        •
        •
       break;

   case OBJ_HIC_MBAR:  /* M_Bar */
        •
        •
       break;

   default:
       break;

        }

    }

}
```

- Calculator functions -

[Function name] `LibCalBuf2Dat`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCalBuf2Dat(byte *c_dat,const byte *c_buf)
```

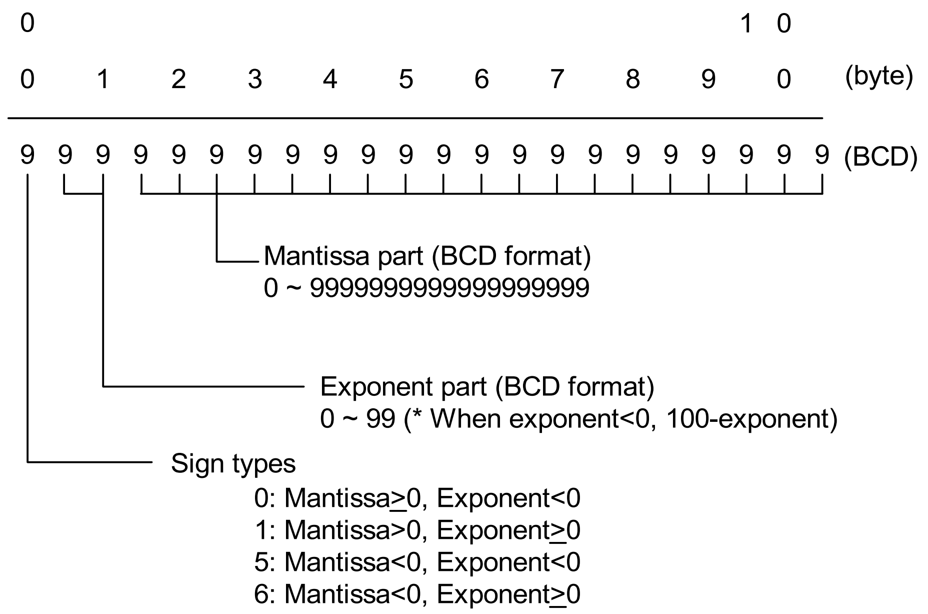
[Arguments]

```
byte        *c_dat :OUT  Operation buffer (Exponential format) [CAL_EZSIZE]
const byte  *c_buf :IN   Operation buffer (Display format) [CAL_BUFSIZE]
```

[Return values] `None`

[Description] Converts the display format operation buffer of the Calculator into exponential format.

[Supplement] The format of c_dat[CAL_EZSIZE] is as follow:


0 1 2 3 4 5 6 7 8 9 10 (byte)
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 (BCD)
Mantissa part (BCD format)
0 ~ 99999999999999999
Exponent part (BCD format)
0 ~ 99 (* When exponent<0, 100-exponent)
Sign types
0: Mantissa≥0, Exponent<0
1: Mantissa>0, Exponent≥0
5: Mantissa<0, Exponent<0
6: Mantissa<0, Exponent≥0

[Example]

```
[  0.]
  →{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}

[          3.14]
  →{0x10,0x03,0x14,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}

[-123456789012.]
  →{0x61,0x11,0x23,0x45,0x67,0x89,0x01,0x20,0x00,0x00,0x00}

[ 1.41421356237]
  →{0x10,0x01,0x41,0x42,0x13,0x56,0x23,0x70,0x00,0x00,0x00}
```

```
[ 0.00006352117]
  →{0x09,0x56,0x35,0x21,0x17,0x00,0x00,0x00,0x00,0x00,0x00}

[-          0.2]
  →{0x59,0x92,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
```

- Calculator functions -

[Function name] `LibCalDat2Buf`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibCalDat2Buf(byte *c_buf,const byte *c_dat)
```

[Arguments]

```
byte       *c_buf :OUT  Operation buffer(Display format)[CAL_BUFSIZE]
const byte *c_dat :IN Operation buffer(Exponential format)[CAL_EZSIZE]
```

[Return values] `None`

[Description] Converts the exponential format operation buffer of the Calculator into display format.

- Debug functions -

[Function name] `LibPutMsgDlg`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibPutMsgDlg(byte *format, ...)
```

[Arguments]

```
byte *format       :IN     Formatted string
```

[Return values] `None`

[Description] Displays a string formatted in conformity with printf().
Performs a dialog display using the characters specified by "format" based on the conversion specification of existing control character string, and waits for touching.
The escape sequence in the window is '\r'.

[Note] This function cannot be used in other bank libraries since BSS is used.

[Examples of usage]

```
int     i;
byte    Buf[40+1];

i = 10;  /* Substitutes a value. */
strcpy(Buf,"ABCDEFGH");  /* Substitutes a value. */
LibPutMsgDlg("buf->[%s]\r[%d]\r[%x]",Buf,i,i); /* Confirms a value. */
```

- Execution result -

```
buf->[ABCDEFGH]
 [10]
 [a]
```

- Debug functions -

[Function name] `LibPutMsgDlg2`

[Syntax]

```
    #include    "define.h"
    #include    "libc.h"
    void LibPutMsgDlg2(byte *format, ...)
```

[Arguments]

```
    byte *format                 :IN     Formatted string
```

[Return values] `None`

[Description] Displays a string formatted in conformity with printf().
Performs a dialog display using the characters specified by "format" based on the conversion specification of existing control character string, and waits for 0.125 seconds before closing the dialog.

- Debug functions -

[Function name] `LibPutMsgDlg3`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibPutMsgDlg3(byte *format, ...)
```

[Arguments]

```
byte *format       :IN     Formatted strings
```

[Return values] `None`

[Description] Displays a string formatted in conformity with printf().
Performs a dialog display using the characters specified by "format" based on the conversion specification of existing control character string, and waits for about 0.5 seconds before closing the dialog.

- Debug functions -

[Function name] `LibPutMsgDlg4`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
void LibPutMsgDlg4(byte *format, ...)
```

[Arguments]

```
byte *format       :IN     Formatted strings
```

[Return values] `None`

[Description] Displays a string formatted in conformity with printf().
Performs a dialog display using the characters specified by "format" based on the conversion specification of existing control character string, and closes the dialog immediately.

- ADDIN functions -

[Function name] `LibExeAddin`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
word LibExeAddin(void)
```

[Arguments] `None`

[Return values]

```
word     Result of execution
         IX_ADIN_SUCCESS    :Success.
         IX_ADIN_COMMERR    :Time-out or Communication failure.
         IX_ADIN_DETECTBLD  :Detected BLD1(low-battery).
         IX_ADIN_DATAFULL   :User data area full.
```

[Description] Not created in a library. This processing is NOP.

- ADDIN functions -

[Function name] `LibGetDLAllNum`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
word LibGetDLAllNum(byte part)
```

[Arguments]

```
byte      part      :IN       IB_DLALL_COUNT  (Program AND Data)
                              IB_DLPROG_COUNT (Program Only)
                              IB_DLDATA_COUNT (Data Only)
```

[Return values]

```
word      Total number(0= not download)
```

[Description] Gets total number of download program and data.

- ADDIN functions -

[Function name] `LibGetUserMode`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
void LibGetUserMode(word *mode_code, word *status, byte condition);
```

[Arguments]

```
word      *mode_code         :OUT Mode Code
word      *status            :OUT Status
byte      condition          :IN  IB_DLFIRST_SRCH (first search)
                                  IB_DLNEXT_SRCH  (next search)
```

[Return values] `None`

[Description] Gets mode-code and status. The search ends if mode_code is 0xffff.
Do the next search after the first search.

[Note]

[Examples of usage]

```
word     mode_code;
word     status;
byte     condition;

condition=IB_DLFIRST_SRCH;
do{
        LibGetUserMode(&mode_code,&status,condition);
        if(mode_code==0xffff)     break;
        condition=IB_DLNEXT_SRCH;
                :
                :
}while(1);
```

- ADDIN functions -

[Function name] `LibGetProgramName`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
bool LibGetProgramName(byte *name_str, byte main_code, byte sub_code);
```

[Arguments]

```
byte     *name_str          :OUT Pointer of program name buffer(16bytes)
byte     main_code          :IN  Main mode Code
byte     sub_code           :IN  Sub mode code
```

[Return values]

```
bool     Results            :TRUE     Success
                            :FALSE    Not corresponding
```

[Description] Gets the program name of the specified mode.
All names of the OS program are output with ffh.
Name buffer is cleared in NULL when there is no specified mode.

[Note] Prepare the array which stores a program name.

[Examples of usage]

```
byte    name[16];
bool    judge;
byte    sub_mode;

sub_mode=0x01;
judge= LibGetProgramName(name,IB_MADDIN,sub_mode);
```

- ADDIN functions -

[Function name] `LibGetLibVer`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
bool LibGetLibVer(byte *ver_str, byte main_code, byte sub_code);
```

[Arguments]

byte	*ver_str	:OUT Pointer of library version buffer(17bytes)
byte	main_code	:IN Main mode Code
byte	sub_code	:IN Sub mode code

[Return values]

bool	Results	:TRUE	Success
		:FALSE	Not corresponding

[Description] Gets the library version which was linked together in the specified mode.
Version buffer is cleared in NULL when there is no specified mode.

String format:
Example) ”1999101015300100”
19991018=Date (=1999/10/18)
1530=Time (=15:30)
0100=Version number (=1.00)

[Note] Prepare the array which stores a version character string.

[Examples of usage]
```
byte    ver[17];
bool    judge;
byte    sub_mode;

sub_mode=0x01;
judge=LibGetLibVer(ver,IB_MADDIN,sub_mode);
```

- ADDIN functions -

[Function name] `LibGetMenuIcon`

[Syntax]
```
    #include    "define.h"
    #include    "libc.h"
    #include    "l_define.h"
    #include    "l_libc.h"
    bool LibGetMenuIcon(byte far **graph_addr, byte main_code, byte sub_code);
```

[Arguments]
```
    byte     far **graph_addr   :OUT Pointer of menu icon graphics
    byte     main_code          :IN  Main mode Code
    byte     sub_code           :IN  Sub mode code
```

[Return values]
```
    bool     Results            :TRUE    Success
                                :FALSE   Not corresponding
```

[Description] Gets icon graphics.

Graphics format:
```
    {
            X dot size(word) , Y dot size(word),
            data of  1 line(byte),
            data of  2 line(byte),
            :
            data of  n line(byte)
    }
```

[Examples of usage]
```
    byte    far *graph_addr;
    bool    judge;

    sub_mode=0x01;
    judge=LibGetMenuIcon(&graph_addr,IB_MADDIN,sub_mode);
    LibPutGraph(5,5,graph_addr);
    LibPutDisp();
```

- ADDIN functions -

[Function name] `LibGetListIcon`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
bool LibGetListIcon(byte far **graph_addr, byte main_code, byte sub_code);
```

[Arguments]

```
byte      far **graph_addr   :OUT Pointer of list icon graphics
byte      main_code          :IN  Main mode Code
byte      sub_code           :IN  Sub mode code
```

[Return values]

```
bool      Results            :TRUE     Success
                             :FALSE    Not corresponding
```

[Description] Gets icon graphics.

Graphics format:

```
{
        X dot size(word) , Y dot size(word),
        data of 1 line(byte),
        data of 2 line(byte),
        :
        data of n line(byte)
}
```

[Examples of usage]

```
byte    far *graph_addr;
bool    judge;

sub_mode=0x01;
judge=LibGetListIcon(&graph_addr,IB_MADDIN,sub_mode);
LibPutGraph(5,5,graph_addr);
LibPutDisp();
```

- ADDIN functions -

[Function name] `LibCheckPMode`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
#include    "l_define.h"
#include    "l_libc.h"
bool LibCheckPMode(byte main_code, byte sub_code, word status);
```

[Arguments]

```
byte      main_code          :IN  Main mode Code
byte      sub_code           :IN  Sub mode code
word      status             :IN  Program status
```

[Return values]

```
bool      Results            :TRUE    Corresponding
                             :FALSE   Not corresponding
```

[Description] It checks whether there is a program concerning specified mode and status.

- ADDIN functions -

[Function name] `LibSubEntrySave`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibSubEntrySave(byte *name_str, byte *sub_entry);
```

[Arguments]

```
byte      *name_str          :IN  Pointer of registration file name
byte      *sub_entry         :OUT Sub-Entry number of registered file name
```

[Return values]

```
bool      Results            :TRUE     Success
                             :FALSE    Error
```

[Description] Registration of the data file name.
The state is output to "SubEntryStat". "SubEntryStat" is global variables.
"sub_entry" is undecided when the return value is a error.
When generating a file in the "Addin" program, write a generated file name with the document and so on.

[Note]

[Registration form of data file name]

Format is 15bytes character string and 00h.

| character string | + | 00h |
|<- 15 bytes ->|

[About “SubEntryStat”]

The following status code is output to "SubEntryStat".

IB_SERR_RNEW	:New registration (Success)
IB_SERR_RALDY	:Already registration (Success)
IB_SERR_FOPEN	:Illegal file name (Error)
IB_SERR_NOSUBC	:No Sub-Entry (Error)
IB_SERR_NGSUBC	:Sub-Entry number is outside of the range (Error)
IB_SERR_ALDYFL	:Already used file name (Error)
IB_SERR_NOFILE	: The file name is not registered (Error)
IB_SERR_INJUST	:Illegal input condition (Error)

[Examples of usage]

```
byte    entry_num;
bool    judge;

judge= LibSubEntrySave(“TEST”,&entry_num);
```

- ADDIN functions -

[Function name] `LibSubEntryDel`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibSubEntryDel(byte *name_str, byte *sub_entry);
```

[Arguments]

```
byte      *name_str          :IN  Pointer of deleting file name
byte      *sub_entry         :OUT Deleted Sub-Entry number
```

[Return values]

```
bool      Results            :TRUE     Success
                             :FALSE    Error
```

[Description] It deletes the registered file name, and deletes all data related to Sub-Entry.
The data which was registered using the "Sub-Entry" number is deleted.
The state is output to "SubEntryStat". "SubEntryStat" is global variables.
"sub_entry" is undecided when the return value is a error.

[Examples of usage]

```
byte    entry_num;
bool    judge;

judge= LibSubEntryDel(“TEST”,&entry_num);
```

- ADDIN functions -

[Function name] `LibSubEntryRename`

[Syntax]
```
    #include    "define.h"
    #include    "libc.h"
    bool LibSubEntryRename(byte *old_name, byte *new_name, byte *sub_entry);
```

[Arguments]
```
    byte     *old_name          :IN  Pointer of old file name
    byte     *new_name          :IN  Pointer of new file name
    byte     *sub_entry         :OUT Renamed Sub-Entry number
```

[Return values]
```
    bool     Results            :TRUE    Success
                                :FALSE   Error
```

[Description] It renames a registered file name.
The state is output to "SubEntryStat". "SubEntryStat" is global variables.
"sub_entry" is undecided when the return value is a error.

[Examples of usage]
```
      byte    entry_num;
      bool    judge;

      judge= LibSubEntryDel("TEST","SAMPLE",&entry_num);
```

- ADDIN functions -

[Function name] `LibSubEntrySearch`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
bool LibSubEntrySearch(byte *name_str, byte *sub_entry);
```

[Arguments]
```
byte      *name_str          :IN  Pointer of search file name
byte      *sub_entry         :OUT Corresponded Sub-Entry number
```

[Return values]
```
bool      Results            :TRUE     Success
                             :FALSE    Error
```

[Description] Gets Sub-Entry number from the registered file name.
The state is output to "SubEntryStat". "SubEntryStat" is global variables.
"sub_entry" is undecided when the return value is a error.

[Examples of usage]
```
byte    entry_num;
bool    judge;

judge= LibSubEntrySearch(〝TEST〞,&entry_num);
```

- ADDIN functions -

[Function name] `LibGetSubEntrySt`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibGetSubEntrySt(byte *name_str, byte sub_entry);
```

[Arguments]

```
byte      *name_str           :OUT Pointer of file name buffer(16bytes)
byte      sub_entry           :IN  Search Sub-Entry number
```

[Return values]

```
bool      Results             :TRUE     Success
                              :FALSE    Error
```

[Description] Gets file name from Sub-Entry number.
File name buffer is cleared in NULL when the return value is a error.
The state is output to "SubEntryStat". "SubEntryStat" is global variables.
Don't specify sub_entry = 0x00.

[Note] Prepare the array which stores a file name.

[Examples of usage]

```
byte      fname[16];
byte      entry_num;
bool      judge;

entry_num=0x01;
judge= LibSubEntrySearch(fname,entry_num);
```

- ADDIN functions -

[Function name] `LibGetSubEntNum`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibGetSubEntNum(void);
```

[Arguments] none

[Return values]

```
word     Results            :Total number
```

[Description] Gets total number of the registered file name(Sub-Entry number).

- ADDIN functions -

[Function name] `LibGetAllEntry`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
bool LibGetAllEntry(byte *name_str, byte *main_entry, byte *sub_entry);
```

[Arguments]

```
byte     *name_str          :IN  Pointer of file name buffer(15+1bytes)
byte     *main_entry        :OUT main entry code
byte     *sub_entry         :OUT sub entry code
```

[Return values]

```
bool     Results            :TRUE    Success
                            :FALSE   Error
```

[Description] Gets main entry code and sub entry code.
The state is output to "SubEntryStat". "SubEntryStat" is global variables.

- Binary file access functions -

[Function name] `LibUbfFindFirst`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
int LibUbfFindFirst(byte *bfnm, ubf_t *buf)
```

[Arguments]

```
byte    *bfnm         :IN    Search filter (Max==MAX_FILENAMELEN)
ubf_t   *buf          :OUT   Struct for file search
```

[Return values]

```
Search handle      Positive value (0-3)      : When applicable file found
                   Error code (negative value):
                                Error (including when file not found)
```

[Description] Search for a binary file (the first time only).
When this finds an applicable file, it writes out the filename and file size, and returns a search handle (simultaneous search is possible for up to four types).
* When search is finished, be sure to execute LibUbfFindClose.
Struct for search "ubf_t"

```
typedef struct {
    int   size;                          /* File size (number of bytes) */
    byte  name[MAX_FILENAMELEN+1];       /* Filename */
}ubf_t;
```

[Examples of usage]

```
if( (findh=LibUbfFindFirst((byte *)"*.txt",buf)) >= 0){
   while( ret>=0 ){
      ret = LibUbfFindNext(findh,buf);
   }
   LibUbfFindClose(findf);
}
```

- Binary file access functions -

[Function name] `LibUbfFindNext`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
int LibUbfFindNext(int sh, ubf_t *buf)
```

[Arguments]

```
    int     sh            :IN    Search handle (return value of LibUbfFindFirst)
    ubf_t   *buf          :OUT   Struct for file search
```

[Return values]	Result	0	: When applicable file found
		Error code (negative value)	: Error (including when file not found)

[Description] Search for a binary file (from the second time onward).
This searches for a file that has conditions specified by LibUbfFindFirst and, when it finds an applicable file, it writes out the filename and file size and returns 0.
* When search is finished, be sure to execute LibUbfFindClose.

[Examples of usage] Refer to "LibUbfFindFirst".

- Binary file access functions -

[Function name] `LibUbfFindClose`

[Syntax]
```
  #include    "define.h"
  #include    "libc.h"
  int LibUbfFindClose(int sh)
```

[Arguments]
```
     int     sh  :IN      Search handle (return value of LibUbfFindFirst)
```

[Return values]
```
     Result      0                           : Normal end
                 Error code (negative value): Error
```

[Description] Close the search for binary files.

[Examples of usage] Refer to "LibUbfFindFirst".

- Binary file access functions -

[Function name] `LibUbfOpen`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
int LibUbfOpen(byte *bfnm, int opnmd)
```

[Arguments]
```
byte *bfnm  :IN      Filename
int  opnmd  :IN      File open mode
                 _UBFOPEN_READ              Read only
                 _UBFOPEN_READ_SHARE        Read only (sharing enabled)
                 _UBFOPEN_WRITE             Write only
                 _UBFOPEN_READWRITE         Read and write
                 _UBFOPEN_READWRITE_SHARE   Read and write (sharing enabled)
```

[Return values]
```
File handle        Positive value (0-15)     : Normal end
                   Error code (negative value): Error
```

[Description] Open a binary file.
This opens a binary file in the mode specified by "opnmd" and returns the file handle.
Up to 16 files can be opened simultaneously.
The file pointer immediately after a file is opened indicates the top of the file.

[Examples of usage]
```
/* Open source file */
if( (fh1=LibUbfOpen((byte *)"source.txt",_UBFOPEN_READWRITE)) >= 0 ){
   /* Open destination file */
   if( (fh2=LibUbfOpen((byte *)"dest.txt",_UBFOPEN_READWRITE)) >= 0 ){
      /* Copy until read size is 0 */
      while(1){
         if( (ret=LibUbfRead(fh1,buf,sizeof(buf))) > 0 ){    /* Read */
            size = ret;
            ret = LibUbfWrite(fh2,buf,size);    /* Write */
         }
         if( ret <= 0 )
            break;
      }
      LibUbfClose(fh2);       /* Close destination file */
   }
   LibUbfClose(fh1);          /* Close source file */
}
```

- Binary file access functions -

[Function name] `LibUbfWrite`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
int LibUbfWrite(int fd, void *buf, int sz)
```

[Arguments]

```
int   fd    :IN      File handle
void  *buf  :IN      Written data
int   sz    :IN      Number of bytes written
```

[Return values]

```
Number of bytes written     Positive value: Normal end
                            Error code (negative value): Error
```

[Description] Write data to a binary file (add to the end of the file).
The file pointer moves to the end after the write operation is finished.

[Examples of usage] Refer to "LibUbfOpen".

- Binary file access functions -

[Function name] `LibUbfRead`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
int LibUbfRead(int fd, void *buf, int sz)
```

[Arguments]

int	fd	:IN	File handle
void	*buf	:OUT	Read buffer
int	sz	:IN	Number of bytes read

[Return values]

Number of bytes actually read	Positive value: Normal end
	Error code (negative value): Error

[Description] Read data from a binary file.
This reads data from the position of file pointer and moves back by the amount it has read the file pointer.

[Examples of usage] Refer to "LibUbfOpen".

- Binary file access functions -

[Function name] `LibUbfSeek`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
int LibUbfSeek(int fd, int pos)
```

[Arguments]
```
int     fd  :IN      File handle
int     pos :IN      Seek position of file pointer
```

[Return values]
```
Result       0     : Normal end
             Error code (negative value): Error
```

[Description] Move the file pointer to the specified position.
Specifying a negative value or a value larger than the file size results in error.

[Examples of usage]
```
pos = 100;
if( LibUbfSeek(fh,pos)==0 ){
  readsize = LibUbfRead(fh,buf,sizeof(buf));
}
```

- Binary file access functions -

[Function name] `LibUbfClose`

[Syntax]

```
  #include    "define.h"
  #include    "libc.h"
  int LibUbfClose(int fd)
```

[Arguments]

```
     int fd      :IN      File handle
```

[Return values]

```
     Result       0     : Normal end
               Error code (negative value): Error
```

[Description] Close a binary file.

[Examples of usage] Refer to "LibUbfOpen".

- Binary file access functions -

[Function name] `LibUbfRemove`

[Syntax]
```
 #include    "define.h"
 #include    "libc.h"
 int LibUbfRemove(byte *bfnm)
```

[Arguments]
```
    byte    *bfnm        :IN    Filename
```

[Return values]
```
    Result       0     : Normal end
                 Error code (negative value): Error
```

[Description] Remove a binary file.
However, a file currently used cannot be removed.

[Examples of usage]
```
ret = LibUbfRemove("test.txt");
```

- Binary file access functions -

[Function name] `LibUbfRename`

[Syntax]
```
  #include    "define.h"
  #include    "libc.h"
  int LibUbfRename(byte *oldbfnm, byte *newbfnm)
```

[Arguments]
```
    byte    *oldbfnm      :IN    Filename before rename
    byte    *newbfnm      :IN    Filename after rename
```

[Return values]
```
    Result       0      : Normal end
              Error code (negative value): Error
```

[Description] Rename a binary file.
However, a file currently used cannot be renamed.

[Examples of usage]
```
ret = LibUbfRename("test.txt","test.bak");
```

- Binary file access functions -

[Function name] `LibUbfLength`

[Syntax]

```
 #include    "define.h"
 #include    "libc.h"
int LibUbfLength(int fd)
```

[Arguments]

```
    int   fd    :IN       File handle
```

[Return values]

Number of bytes of target file	Positive value	: Normal end
	Error code (negative value): Error	

[Description] Get the size of a binary file currently opened.
* If you want to get the size of a currently closed file, use LibUbfFindFirst.

[Examples of usage]

```
if( (fh=LibUbfOpen((byte *)"test.txt",_UBFOPEN_READWRITE)) >= 0 ){
   size = LibUbfLength(fh);
   LibUbfClose(fh);
}
```

- Binary file access functions -

[Function name] `LibUbfFlush`

[Syntax]

```
#include     "define.h"
#include     "libc.h"
int LibUbfFlush(int fd)
```

[Arguments]

```
    int      fd   :IN        File
```

[Return values]

Result	0	: Normal end
	Error code (negative value):	Error

[Description] Write data gathered in the buffer onto the memory for a binary file that is currently written.

[Examples of usage]

```
for( ct=0;; ct++ ){
   /* Copy until read size is 0 */
   if( (ret=LibUbfRead(fh1,buf,sizeof(buf))) > 0 ){    /* Read */
      size = ret;
      if( (ret=LibUbfWrite(fh2,buf,size)) > 0 ){      /* Write */
         if( (ct%3)==2 ){    /* Flush every three times */
            ret = LibUbfFlush(fh2);
         }
      }
   }else{
      break;
   }
   if( ret<0 )
      break;
}
```

- Binary file access functions -

[Function name] `LibUbfGetFree`

[Syntax]

```
 #include    "define.h"
 #include    "libc.h"
int LibUbfGetFree(void)
```

[Arguments] None

[Return values]

Result	Free memory space (number of bytes): Normal end
	Error code (negative value): Error

[Description] Get the current free memory space.

[Examples of usage]

```
if( (ret=LibUbfGetFree())>=0 ){
   freesize = ret;
}
```

- Error codes output only by binary file related libraries –

Binary file related libraries ("LibUbf...") output one of their own error codes when an error occurs. The error codes and their descriptions are as follows:

_UBFERR_MEMORY	: No free space
_UBFERR_FILENAME	: Illegal filename
_UBFERR_PARAMETER	: Illegal parameter
_UBFERR_SYSTEM	: System abnormality
_UBFERR_MAXFILES	: Maximum number of files exceeded
_UBFERR_ACCESS	: File access disabled
_UBFERR_FILETYPE	: Not a binary file
_UBFERR_FILTER	: Illegal search condition
_UBFERR_NOTFIND	: Search closed
_UBFERR_FINDHANDLE	: No free search handle
_UBFERR_SEEKPOS	: Abnormal seek position
_UBFERR_OTHER	: Other errors

- Serial/USB Communication functions -

[Function name] `LibSrlPortOpen`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlPortOpen(SRL_STAT *po);
```

[Arguments]

```
SRL_STAT  *po       :IN      Pointer of- Serial/USB communication status
```

[Return values]

```
word     err_code           :IW_SRL_NOERR    No error
                            :IW_SRL_PRMERR   Parameter error
```

[Description] It opens the communication port ,and enables the sending and the receiving.
The communication status is following structure.

[Note] Setting according to the argument is not reflected in the communication for USB. The communication is done in a high-speed mode by the bulk transfer.

```
typedef struct SRL_STAT {
       byte   port;              /* Port number */
       byte   speed;             /* BPS */
       byte   parit;             /* Parity bit */
       byte   datab;             /* Data bit length */
       byte   stopb;             /* Stop bit length */
       byte   fctrl;             /* Flow control */
} SRL_STAT;
```

<Port number>

IB_SRL_COM2	: 9 pin serial
IB_SRL_COM3	: USB

<BPS>

IB_SRL_300BPS	: 300 bps
IB_SRL_600BPS	: 600 bps
IB_SRL_1200BPS	: 1200 bps
IB_SRL_2400BPS	: 2400 bps
IB_SRL_4800BPS	: 4800 bps
IB_SRL_9600BPS	: 9600 bps
IB_SRL_19200BPS	: 19200 bps
IB_SRL_38400BPS	: 38400 bps
IB_SRL_57600BPS	: 57600 bps
IB_SRL_115200BPS	: 115200 bps

<Parity bit>

IX_SRL_NONE	: NONE parity
IX_SRL_ODD	: ODD parity
IX_SRL_EVEN	: EVEN parity

<Data bit length>

IX_SRL_7DATA	: 7 bits length
IX_SRL_8DATA	: 8 bits length

<Stop bit length >

IX_SRL_1STOP	: 1 stop bit
IX_SRL_2STOP	: 2 stop bit

<Flow control>

IX_SRL_NOFLOW	: No control
IX_SRL_RSCS	: RS/CS control
IX_SRL_XONOFF	: XON/XOFF & RS/CS control
IX_SRL_XONOFFONLY	: XON/XOFF control

[Examples of usage]

```
SRL_STAT        srl_status;
word            err;

srl_status.port=IB_SRL_COM2;
srl_status.speed=IB_19200BPS;
srl_status.parit=IX_SRL_NONE;
srl_status.datab=IX_SRL_7DATA;
srl_status.stopb=IX_SRL_2STOP;
srl_status.fctrl=IX_SRL_RSCS;

err= LibSrlPortOpen (&srl_status);
```

-- Serial/USB Communication functions -

[Function name] `LibSrlPortClose`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlPortClose(void);
```

[Arguments]

None

[Return values]

```
word     err_code           :IW_SRL_NOERR    No error
                            :IW_SRL_CLSERR   Not Closesed
```

[Description] It closes the communication port.
The error occurs when data remains in the transmission buffer or the transmission register.

-- Serial/USB Communication functions -

[Function name] `LibSrlPortFClose`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlPortFClose(void);
```

[Arguments]

```
None
```

[Return values]

```
word     err_code          :IW_SRL_NOERR    No error
```

[Description] It closes a communication port compulsorily.
It closes even if data remains in the transmission register or in the transmission buffer.
At this time, a transmission buffer is cleared and the last sending character has the possibility of the mis-conversion.
Because it closes regardless of the communication condition, generally, use "LibSrlPortClose".

-- Serial/USB Communication functions -

[Function name] `LibSrlRxBufClr`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlRxBufClr(void);
```

[Arguments]

None

[Return values]

```
word     err_code           :IW_SRL_NOERR    No error
```

[Description] It clears a receiving buffer.

-- Serial/USB Communication functions -

[Function name] `LibSrlTxBufClr`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlTxBufClr(void);
```

[Arguments]

None

[Return values]

```
word     err_code           :IW_SRL_NOERR   No error
```

[Description] It clears a sending buffer.

-- Serial/USB Communication functions -

[Function name] `LibSrlGetDteStat`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlGetDteStat(word *num, word *flag);
```

[Arguments]

```
word     *num             :OUT Number of data in receiving buffer
word     *flag            :OUT Status flags
                          IX_SRL_OV       Buffer over flow
                          IX_SRL_CB       DCE busy
                          IX_SRL_TB       DTE busy
                          IX_SRL_OE       Over run error
                          IX_SRL_PE       Parity error
                          IX_SRL_FE       Framing error
```

[Return values]

```
word     err_code         :IW_SRL_NOERR   No error
                          :IW_SRL_RCVERR  Receive error
```

[Description] Gets DTE status.

[Examples of usage]

```
word    err;
word    rcv_num;
word    flag;

err=LibSrlGetDteStat(&rcv_num,&flag);
if(err==IW_SRL_RCVERR){
        if((flag&IX_SRL_OV)!=0){
                :
        }else if((flag&IX_SRL_OE)!=0){
                :
        }else if(

}
```

-- Serial/USB Communication functions -

[Function name] `LibSrl232CStat`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
byte LibSrl232CStat(void);
```

[Arguments]

```
None
```

[Return values]

```
byte      flag                :IX_SRL_ER       ER
                              :IX_SRL_RS       RS
                              :IX_SRL_CS       CS
                              :IX_SRL_CD       CD
                              :IX_SRL_DR       DR
```

[Description] Gets status of RS232C signal line.

[Note] They return the state of the memorized signal line for USB.
As CS, DR, and CD signal line, the value of '1' is always returned.

[Examples of usage]

```
word    flag;

flag=LibSrl232CStat();
if((flag&IX_SRL_DR)!=0){          /* DR ON */
        :
}
```

-- Serial/USB Communication functions -

[Function name] `LibSrlRateSet`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlRateSet(byte speed);
```

[Arguments]

```
byte      speed              :IN  BPS
```

[Return values]

```
word      err_code           :IW_SRL_NOERR    No error
                             :IW_SRL_PRMERR   Parameter error
```

[Description] Changes communication speed of DTE.

-- Serial/USB Communication functions -

[Function name] `LibSrlGetTBufSpace`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlGetTBufSpace(void);
```

[Arguments]

```
None
```

[Return values]

```
word     space              :Number of empty characters
```

[Description] Gets number of empty characters in sending buffer.

-- Serial/USB Communication functions -

[Function name] `LibSrlSendByte`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlSendByte(byte tmode, byte data);
```

[Arguments]

```
byte     tmode              :IN  Sending mode
                                 IB_FOLLOW_BUSY
                                 IB_IGNORE_BUSY
byte     data               :IN  Sending data
```

[Return values]

```
word     err_code           :IW_SRL_NOERR    No error
                            :IW_SRL_TRSERR   Sending not possible
```

[Description] Sends one data.
When “IW_IGNORE_BUSY" is specified, data is written directly into the transmission register.
Please specify “IB_FOLLOW_BUSY" usually.

-- Serial/USB Communication functions -

[Function name] `LibSrlRecvByte`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlRecvByte(byte *data);
```

[Arguments]

```
byte     *data              :IN  Receive data
```

[Return values]

```
word     err_code           :IW_SRL_NOERR    No error
                            :IW_SRL_NODATA   No receive data
```

[Description] Gets one data in receiving buffer.

-- Serial/USB Communication functions -

[Function name] `LibSrlPreRead`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlPreRead(word oft, byte *data);
```

[Arguments]

```
word     oft                :IN  Buffer offset
byte     *data              :OUT Read data
```

[Return values]

```
word     err_code           :IW_SRL_NOERR    No error
                            :IW_SRL_NODATA   No receive data
```

[Description] Gets one data in receiving buffer (no pointer update).
Please adjust “oft” to one when you read the next data.

-- Serial/USB Communication functions -

[Function name] `LibSrlSendBreak`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlSendBreak(byte time);
```

[Arguments]

```
byte      time                :IN  Signal length(1=100ms)
                                        "time" from 1 to 9
```

[Return values]

```
word      err_code            :IW_SRL_NOERR    No error
                              :IW_SRL_PRMERR   Parameter error
```

[Description] The break signal is sent during the specified time.
The break signal is inserted between data and data when there is transmission data.

[Note] The interruption signal is not transmitted for USB. The return value returns only ″IMW_COMM_NOERR″.

-- Serial/USB Communication functions -

[Function name] `LibSrlSendBlock`

[Syntax]

```
#include    "define.h"
#include    "libc.h"
word LibSrlSendBlock(byte *data, word size);
```

[Arguments]

byte	*data	:IN Pointer of buffer which stored sending data
word	size	:IN Sending size(byte)

[Return values]

word	err_code	:IW_SRL_NOERR	No error
		:IW_SRL_TRSERR	Empty lack of sending buffer

[Description] Sends a block data.
"IW_SRL_TRSERR" occurs when specified data isn't stored in the sending buffer.
Therefore, you must wait until being possible to transmit or transmit data by dividing.

-- Serial/USB Communication functions -

[Function name] `LibSrlRecvBlock`

[Syntax]
```
     #include    "define.h"
     #include    "libc.h"
     word LibSrlRecvBlock(byte *data, word size, word *num);
```

[Arguments]
```
     byte      *data               :OUT Pointer of storage buffer
     word      size                :IN  Buffer size (size > 0)
     word      *num                :OUT Number of stored data
```

[Return values]
```
     word      err_code            :IW_SRL_NOERR    No error
                                   :IW_SRL_NODATA   No data error
```

[Description] It reads a block data from the receiving buffer.

[Examples of usage]
```
        byte     buf[1024];
        word     size,num,err;

        size = 1024;
        err = LibSrlRecvBlock(buf,size,&num);
```

-- Serial/USB Communication functions -

[Function name] `LibSrlGetOpenStat`

[Syntax]
```
#include    "define.h"
#include    "libc.h"
word LibSrlGetOpenStat(void);
```

[Arguments]
```
None
```

[Return values]
```
word     open_stat          :IB_NO_OPEN      No open
                            :IB_COM2_OPEN    9pin serial open
                            :IB_COM3_OPEN    USB port open
```

[Description] Gets the open condition of the communication port.