Library Function
For Application Program

Character Input & Drag Event

1999

1. Introduction

Each mode of Digital Diary consists of three states, "Data Input", "List Display", and "Data Display". This character-input library is used for "Data Input" and "Data Display".

2. About Functions

The character-input library is composed of the following function groups. As it is shown in the example below, three functions- "Initialization function", "Character string display function", and "Buffer operation function" - must always be used as a set. Data input and data display functions are provided on "Character string display function" and "Buffer operation function".

"`LibTxtInit()`" Initialization function (Common to data input and display)

This function initializes several variables for text input. This function must be called once when the display contents are loaded into the text buffer after the text area has been specified. (This function does not initialize the contents of the text buffer.)

"`LibTxtInp()`" Buffer operation function (For input.)

This is a routine, which is used to edit the contents in the text buffer. When the function receives a key code from the software keyboard process LibGetKeyM, it starts the text buffer write process. Additionally, the function performs the drag selection process (cut, copy, paste) based on the internal touch waiting. The function also controls the scroll bar operation on the character-input screen.

"`LibTxtDsp()`" Character string display function (For input.)

This is a display routine during text input. This function displays characters, which have been input, and also switches the software keyboard automatically.

"`LibTxtDspS()`" Buffer operation function (For data display)

This function performs the drag (copy) process and scroll bar process for data display.

"LibTxtDspC( )" Character string display function (For data display)

This function is used to display data in each mode. The body is a core routine of the character display function during text input, LibTxtDsp( ).

3. About the Text Input Structure

In this library, several input conditions, such as pointer for the text buffer and display coordinates on the screen are specified in the structure for text input, TXTP.
The text-input structure contains work areas for the character-input process. Therefore, when specifying more than one structure, this makes it possible to have multiple independent text areas within the screen.

When a text display area is declared, a touch area for the character input library is also specified. Great care should be taken.
Actually, proportional characters are displayed in the specified area and character operations other than "Cursor move" and "Drag process (Copy and Paste)" provided by the character-input library become invalid in this area.

The following describes the contents of the structure for text input. All members used as input conditions for text input must be set before calling the "Initialization function (LibTxtInit())".
Initial setting of members defined as text input library work RAM is not needed since they are work areas for the character-input function.

```
/* General-purpose character input/data display parameter information */
        typedef struct  TXTP {

    /* Input conditions text input library  */
        int             st_x;       /*X-coordinate of left end of text area */
        int             ed_x;       /* X-coordinate of right end of text area */
        int             st_y;       /* Y-coordinate of top end of text area */
        int             it_y;       /* Line spacing of text area */
        int             MAXGYO;     /* Maximum number of text display lines */
        word    maxmj;              /* Maximum number of allowable input characters. */
        byte    font;               /* Font size designation */
        byte    csen;               /* Cursor display enabled */   See *1.
        byte    rtnen;              /* CR code display enabled */  See *2.
        byte    *txbf;              /* Text buffer pointer */
        word    *gdcmt;             /* Pointer for guidance comment table */  See *3.
        byte    *gdcmt2;            /* Pointer for guidance comment buffer */  See *3.
        word    txtobj;             /* Object when text area is touched.   */
        word    sbrobj;             /* Object when the scroll bar is touched */
        byte    *kwb_0;             /* Pointer for keyword buffer 0 */   See *4.
        byte    *kwb_1;             /* Pointer for keyword buffer 1 */   See *4.
        byte    *kwb_2;             /* Pointer for keyword buffer 2 */   See *4.
        byte    *kw_dbk;            /* Pointer for keyword screen saving area */  See *4.
```

```
        TCHTBL          *tchtb;         /* Pointer for touch table  */
        T_SCR_POS       *s_pos;         /* Pointer for text and scroll bar information */

/* Text input library work RAM */
        word    cp;                     /* Cursor pointer  */
        int             xp;             /* X-coordinate of display  */
        int             yp;             /* Y-coordinate of display  */
        int             c_xp;           /* X-coordinate of display cursor */
        int             c_yp;           /* Y-coordinate of display cursor */
        word    dlmp[18];               /* Pointer for start character of line */
        byte    ditp[18];               /* Pointer for start character of line */
        word    csln;                   /* Cursor display line position */
        word    mjln;                   /* Character drawing start position */
        word    mjcnt;                  /* Number of characters in text buffer  */
        byte    ipdpst;                 /* Inputting display status flag  */
        byte    txtst;                  /* Text input status flag  */
        word    sr_sp;                  /* Pointer for highlight start position */
        word    sr_ep;                  /* Pointer for highlight end position */
        int             sr_xp;          /* Pointer (X) for highlight end position */
        int             sr_yp;          /* Pointer (Y) for highlight end position */
        word    srln;                   /* Number of highlight lines  */
        bool    selrv;                  /* Selected highlight flag  */
        word    ktyp;                   /* For saving keyboard type */
        byte    citm;                   /* Current cursor item  */
        byte    wdcnt;                  /* Length of word being input */
        byte    kwnmt;                  /* Prospective item displaying flag */
        byte    kwsln;                  /* Prospective item display selection line */
} TXTP;
```

*1 csen

FALSE(0x00): Cursor display disabled(Normally, this is used for data display)
TRUE (0x01): Cursor display enabled (Normally, this is used for data input.)
0x02 : Cursor display enabled && Keyword is registered, Display is allowed.

*2 rtnen

FALSE(0x00): Disables the CR code display. (Normally, this is used for data display)
TRUE (0x01): Enables the CR code display. (Normally, this is used for data input.)
HALF (0x02): Disables the CR code input. (Normally, this is used for one line input.)
0x03: Limits on the number of characters for each item. (This is used for correction of CONTACTS and MEMO item names.)
0x04: Hides the scroll bar display. (This is used for full-screen display of MEMO.)

*3 For details of "gdcmt" and "gdcmt2", see the [About Guidance display function].

*4. They do not need to set if a value other than "0x02" is set in the member "csen" of the text input structure. See the chapter "10. About keyword registration" for more details.

4. About Text Buffer

The end code (0x00) must be put at the end of the text buffer. As a result, the buffer size becomes (the maximum number of characters + 1 byte).
Additionally, items are divided by the next item code (0xfe) if multiple items exist in one text buffer, like in the CONTACTS mode.

5. About Copy, Cut, and Paste

When making a character string highlighted to select it and copying, cutting, or pasting the selected character string, values shown below are set in the members of the structure "txtst" for text input before calling the input main function "LibTxtInp( )". (See the example for more details.)
Additionally, to copy a character string during data display, "txtst" is set to [TXTCOPY] before calling the function, "LibTxtDspS( )".

TXTCOPY:	Copies a highlighted character string selected to the copy buffer.
TXTCUT:	Deletes the selected character string from the text buffer after the copy process has been completed.
TXTPASTE:	Copies the character string in the copy buffer to the cursor position.

6. Basic Example (data input)

The example for performing a new input of data is shown below.
The specification prepared for this example is expected as follow: An icon that shows the transition to the data display is displayed on the screen, and the program steps out when the icon is touched.

```
/*** Designation of text area  ***/
#define   M_ST_X    10      /* st_x */
#define   M_ST_Y    13      /* st_y */
#define   M_ED_X    50      /* ed_x */
#define   M_IT_Y    9       /* it_y */
#define   M_MAXG    5       /* MAXGYO */

 /*** Scroll bar setting ***/
#define   SCR__Y            M_ST_Y              /* Scroll bar position  */
#define   SCR__SIZE         M_IT_Y*M_MAXG       /* Height of scroll bar  */
#define   SCR__DSP          M_MAXG              /* Number of display records */

byte      mtxbf[2049];              /* Buffer for text input  */
```

```
        TCHTBL InptTch[3];                  /* Text touch area table  */

         /* Keyword saving buffer  */
        byte        keywd_0[33];
        byte        keywd_1[33];
        byte        keywd_2[33];
        byte        dspbk[400];

/*********************************************************************
        [Title]
        New data input (Keyword registration is provided.)
*********************************************************************/
void NewInp(void)
{
        byte kycd;
        TCHSTS          tsts;
        TXTP            m_in_tp;        /* Declaration of structure for text input*/
        T_SCR_POS       m_in_scb;       /* Scroll bar position information */

/*** Text input setting  ***/
        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        = 0x02;   /* Cursor display enabled(Keyword registration enabled) */
        m_in_tp.rtnen       = TRUE;       /* CR code display enabled  */
        m_in_tp.maxmj       = 2048;       /* Maximum number of allowable input characters. */
        m_in_tp.txbf        = mtxbf;      /* Designation of text buffer address  */
        m_in_tp.gdcmt       = telgd;      /* Guidance comment table  */
        m_in_tp.txtobj      = OBJ_TXTTCH;   /* Object when text area is touched.  */
        m_in_tp.sbrobj      = OBJ_SCR_BAR;   /* Object when the scroll bar is touched. */
        m_in_tptp.kwb_0     = keywd_0;    /* Keyword saving buffer 0 */
        m_in_tptp.kwb_1     = keywd_1;    /* Keyword saving buffer 1 */
        m_in_tptp.kwb_2     = keywd_2;    /* Keyword saving buffer 2 */
        m_in_tptp.kw_dbk= dspbk;          /* Keyword display saving area  */
        m_in_tp.tchtb       = InptTch;    /* Pointer for text scroll bar area  */
        m_in_tp.s_pos       = &m_in_scb;  /* Pointer for text and scroll bar information */

        mtxbf[0] = 0;                     /* Initialization of text buffer  */

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

 /*** Text touch area setting ***/
        LibTchStackClr();
```

```
        LibTchStackPush( NULL );
        LibTxtTchSet(&m_in_tp);             /* Text touch area PUSH */

        LibTchStackPush(TchDataKey);        /* Icon touch table for transition of data display */
        LibTchStackPush(TchHardIcon);       /* Hard icon touch table  */

        LibClrDisp();
        LibIconPrint(&TicnDataKey);         /* Icon display for transition of data display */

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

            kycd = LibGetKeyM(&tsts);               /* Waiting for software key  */

            if(kycd==KEY_NONE){

/*** Process when a portion other than the text area, software keys, and scroll bar is touched. ***/
                    if(tsts.obj==OBJ_IC_DATAKEY){       /* When touching "Data display" icon. */
                        if(LibIconClick(&TicnDataKey,&tsts)==TRUE){
                            mem_st = DATA_DSP;          /* To data display  */
                                LibTchInit();
                        }

                    }else if(tsts.obj==OBJ_IC_CLRKEY){      /* "When touching "CLR" icon. */
                        if(LibIconClick(&TicnClrKey,&tsts)==TRUE){
                            TelTextInit();                  /* Initialization of text buffer  */
                                LibTxtInit(&tin_tp); /* Initialization of text input  */
                            }
                    }

    /*** Copy, cut, and paste process  ***/
                    }else if(tsts.obj == OBJ_HIC_CONT){
                        mm = CpMenu();                  /* Copy & paste menu  */
                        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);
            }
            LibTchStackClr();
        }
```

7. Basic Examples (data display)

The following example is for the data display.

```
/*** Designation of text area  ***/
    #define   M_ST_X    10                          /* st_x */
    #define   M_ST_Y    13                          /* st_y */
    #define   M_ED_X    50                          /* ed_x */
    #define   M_IT_Y     9                          /* it_y */
    #define   M_MAXG    6                           /* MAXGYO */

    #define    SCR__Y          M_ST_Y          /* Scroll bar position */
    #define    SCR__SIZE       M_IT_Y*M_MAXG   /* Height of scroll bar */
    #define    SCR__DSP        M_MAXG          /* Number of display records */

    TCHTBL        DdspTch[3];                  /* Text touch area table  */


/************************************
        [Title]
                Data display
************************************/
void DataDsp(void)
{
    TCHSTS      tsts;
    TXTP        m_dd_tp;        /* Declaration of structure for text input  */
    T_SCR_POS   m_dd_scb;       /* Scroll bar position information */

/** 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        = FALSE;        /* 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;            /* Designation of text buffer address  */
        m_dd_tp.txtobj    = OBJ_TXTTCH;   /* Object when text area is touched.  */
        m_dd_tp.sbrobj    = OBJ_SCR_BAR;  /* Object when the scroll bar is touched*/
        m_dd_tp.tchtb     = DdspTch;          /* Pointer for text scroll bar area  */
        m_dd_tp.s_pos     = &m_dd_scb;  /* Pointer for text and scroll bar information */

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

    /*** Text touch area setting ***/
        LibTchStackClr();
        LibTchStackPush( NULL );
        LibTxtTchSet(&m_dd_tp);           /* Text touch area PUSH */

        LibTchStackPush(TchNewKey);   /* Icon touch table for transition of new input.  */
        LibTchStackPush(TchHardIcon);  /* Hard icon touch table  */

        LibClrDisp();
        LibIconPrint(&TicnNewKey);    /* Icon display for transition of new input.  */

        LibPutDisp();

    /*** Data display loop  ***/
        while(mem_st==DATA_DSP){
            if(LibTxtDspC(&m_dd_tp)==TRUE)          /* Text display  */
                LibPutDisp();
            LibTchWait( &tsts );                    /* Waiting for touch  */

            switch(tsts.obj){
                case OBJ_IC_NEWKEY:                 /* New input icon is touched */
                    if(LibIconClick(&TicnNewKey,&tsts)==TRUE){
                        mem_st = NEW_INP;  /* Exit data display loop, and go to new input. */
                            LibTchInit();
                    }
                    break;
                case OBJ_HIC_CONT:
                    if(CpMenu()==1)                 /* Copy & paste menu  */
                        m_dd_tp.txtst = TXTCOPY;    /* Copy */
                    break;
            }

            /*** Text touch process during data display  ***/
            LibTxtDspS(&m_dd_tp,&tsts);
        }
        LibTchStackClr();
    }
```

8. About Guidance Display Function

The “Guidance display” specification that prompts user to input data is now available. However, note that this specification is valid when the text buffer is empty, or when there is a cursor but no text (no character) is available within the item separated by next item code (0xfe).

To use the guidance display function, transferring the pointer for the table containing the comment numbers used for the guidance display to the member, "word *gdcmt", of the text input structure will display the guidance automatically. The number of comment numbers in this table must be matched with that of items in the text buffer (normally, one item in modes other than the TEL mode). (Normally, only one word is used in modes other than the TEL mode.)
At this time, "word *gdcmt2" is ignored and therefore does not need to be specified.

(Examples of usage1)

```
 /*** TEL guidance table  ***/
word telgd[] = {
        213,    /* NAME       */
        202,    /* ADDRESS_(H) */
        203,    /* FAX_(B)    */
        204,    /* PHONE_(B)  */
        205,    /* E-MAIL     */
        206,    /* EMPLOYER   */
        207,    /* FAX_(H)    */
        208,    /* PHONE_(H)  */
        209,    /* MOBILE     */
        210,    /* NOTE       */
};
```

See the chapter "6.Basic examples (data input)" for more about this.

Furthermore, to make the guidance display inactive, "0xffff" is put in the table and the pointer for this table is transferred. To make the guidance display inactive, "0xffff" is put in the table and the pointer for this table is transferred. Additionally, the guidance display does not function during data display. Therefore, it is not necessary to particularly specify "word *gdcmt" during data display.

◆ The guidance display specification for the CONTACTS mode is supported.

To use desired character strings instead of the built-in (fixed) guidance message, the pointer for the table, in which "0xfffe" is put, is specified in "word *gdcmt". Additionally, the pointer for the top of the buffer where a desired character string is input to "word *gdcmt2" is specified.
In the buffer, messages for the number of items is divided by "0xfe" and the end code "0x00" must be put at the end of the messages.

9. About Text Area and Scroll Bar

It is necessary to set up the text area touch table and scroll bar touch table when using the text input function. At the same time, the object codes set to them must be registered to the members (txtobj, sbrobj) of the text-input structure.
This makes it possible to recognize which part of the text area (scroll bar) is touched in the inside of the text input function after it is called many times.

In order to prevent any mismatch between the touch table of the scroll bar area settled and the scroll bar actually displayed, now the function "LibTxtTchSet" that makes it possible to set up the touch table for the text area and the scroll bar area is available. Call this function once after calling the function "LibTxtInit" (text input initialization function).
Additionally, the pointer for the scroll bar position information used for text input (display) must be set in the member "*s_pos" of the text input structure.
Three empty structures (TCHTBL) for the touch table are prepared and the pointers for these structures are set in the member "tchtb" of the text-input structure.

10. About Keyword Registration

ZX483 provides a function that registers and displays keywords when data is input in the scheduler mode.
To use this function, "0x02" is first set in "csen". At the same time, three buffers (33 bytes) for display of prospective keywords are prepared and pointers for these buffers are set in "kwb_0", "kwb_1", and "kwb_2".
Additionally, a buffer with a capacity of 400 bytes for temporary saving of the screen is prepared and its pointer is set in "kw_dbk".
If the keyword registration is not used ("csen" is either "0x00" or "0x01"), it is not necessary to set these pointers.
See the chapter "6. Basic example (data input)" and "7. Basic examples (data display)" for the examples of usage.

<Remark>
There is a specification to perform a keyword registration by "SET" or "ESC" button during data input. To achieve this specification, call the function "LibTxtKeyWordSet(TXTP *tp)" when the button mentioned above is touched.

11. About Current Date/Time Paste Function

The function to insert the current date and time as a character string during data input is available.
To make this function active, the member "byte txtst" of the text input structure is set to "TXTDYTIM". (This is the same as that "txtst" is set to "TXTPASTE" (txtst = TXTPASTE) when making the normal paste function active.)
The current time paste button is to be provided on the input screen of each application, and the application is programmed so that "txtst=TXTDYTIM" is set when such button is touched. (12/24-time format is supported.)

12. About dlmp/ditp

“`dlmp[n]`” and “`ditp[n]`” are provided in the work RAM which is used by the character input library. When referring to them after the key waiting process, it is possible to know which character in the text buffer is currently displayed on each line of the screen or which item is displayed.
This can be used to display the item name during text input or data display.

`word dlmp[n]` Start character position of line "n" on the currently displayed screen. (For example, when 5th character is at the start position, this data is [0x0004].)
If the buffer is empty, "0xffff" is set.

`byte ditp[n]` The number of items in "n" line on the currently displayed screen. (For example, when three items exist, this data is [0x02].)
However, if an item extends over more than one line, this data is [0xff] in lines other than the start line.

dlmp[0] shows the start of the line before the top line displayed on the screen (line currently not displayed).

(Example) TEL mode input screen

("↵" is for CR, and "■" is for a blinking cursor.)

```
0  0xffff   0xff
                              ============================================================
1  0x0000   0x00   |NAME       |S. SUZUKI                                                 |
                              +---------------+------------------------------------------+
2  0x0009   0x01   |EMPLOYER   |CASIO                                                     |
                              +---------------+------------------------------------------+
3  0x000f   0x02   |PHONE      |0123-45-6789                                              |
                              +---------------+------------------------------------------+
4  0x001c   0x03   |FAX        |0123-45-678                                               |
                              +---------------+------------------------------------------+
5  0x0029   0x04   |ADDRESS    |3-2-1 SAKAECHO↵                                           |
                              +---------------+------------------------------------------+
6  0x0037   0xff   |           |HAMURA-SHI↵                                               |
                              +---------------+------------------------------------------+
7  0x0042   0xff   |           |TOKYO, JAPAN                                              |
                              +---------------+------------------------------------------+
8  0x004d   0x05   |E-MAIL     |suzuk■                                                    |
                              +---------------+------------------------------------------+
9  0xffff   0xff   |           |                                                          |
                              ============================================================
10 0xffff   0xff
```

In this example, the contents of the text buffer are as follows:

```
53, 2E, 53, 55, 5A, 55, 4B, 49, FE,
43, 41, 53, 49, 4F, FE,
30, 31, 32, 33, 2D, 34, 35, 2D, 36, 37, 38, 39, FE,
30, 31, 32, 33, 2D, 34, 35, 2D, 36, 37, 38, 39, FE,
33, 2D, 32, 2D, 31, 20, 53, 41, 4B, 41, 45, 43, 48, 4F, 0D,
48, 41, 4D, 55, 52, 41, 2D, 53, 48, 49, 0D,
54, 4F, 4B, 59, 4F, 2C, 4A, 41, 50, 41, 4E, FE,
73, 75, 7A, 75, 6B, 00
```

13. Cautions for Use

(1) To use the above function during data display or correction (copy) data input, the text initialization function "LibTxtInit (or LibTxtDspInit)" must be called after the text data is loaded into the text buffer. If the initialization function is called before loading data into the text buffer in the application program, the initialization function must be called again every time data has been loaded. (No bad impact is expected.)

(2) In the text initialization function, the number of characters in the text buffer is counted. However, if the initialization is made before loading data, the number of characters may become incorrect

depending on the previous contents of the buffer. This may cause incorrect operation such as that dragging cannot be made completely. (Be careful because it looks as if the display is correctly done.)

(3) It is absolutely necessary to define the member "TCHTBL *tchtb /* Pointer for touch table */", one of input conditions for the text input structure. IF THIS IS NOT DEFINED, THE RAM WILL BE DESTROYED. THOUGH IN THE MOST TIME IT WORKS PROPERLY.