For fx-9860G Series

E

fx-9860G Software Development Kit

Libraries

http://world.casio.com/edu/

CASIO®

fx-9860G SDK Libraries

Bdisp_AllClr_DD/ Bdisp_AllClr_VRAM/ Bdisp_AllClr_DDVRAM

These functions clear the whole area of VRAM and/or DD (Display Driver).

```
void Bdisp_AllClr_DD(void);
void Bdisp_AllClr_VRAM(void);
void Bdisp_AllClr_DDVRAM(void);
```

Parameters

Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

Bdisp_AreaClr_DD/ Bdisp_AreaClr_VRAM/ Bdisp_AreaClr_DDVRAM

These functions clear the specified area of VRAM and/or DD (Display Driver).

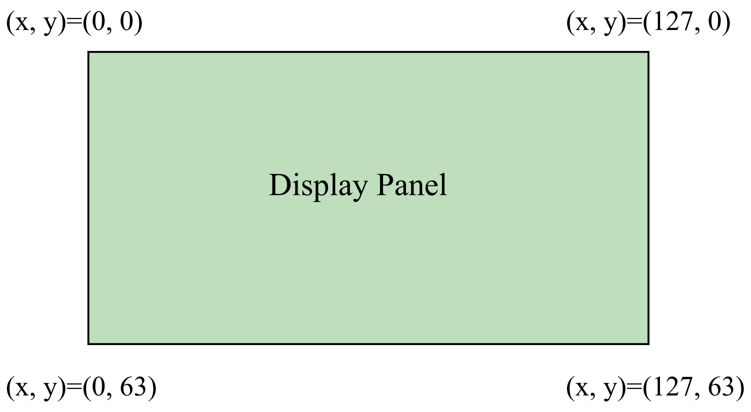
```
void Bdisp_AreaClr_DD(
void Bdisp_AreaClr_VRAM(
void Bdisp_AreaClr_DDVRAM(
    const DISPBOX *pArea     // pointer to structure data
);
```

Parameters

pArea

This is a pointer to the DISPBOX structure. The DISPBOX structure defines a rectangle. The structure specifies the coordinates of two points: the upper-left and lower-right corners of the rectangle. The sides of the rectangle extend from these two points and are parallel to the x-axis and the y-axis.

```
typedef struct tag_DISPBOX{            // Declared in dispbios.h
    int     left;       // 0~127
    int     top;        // 0~63
    int     right;      // 0~127
    int     bottom;     // 0~63
} DISPBOX;
```



Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

Bdisp_AreaReverseVRAM

The Bdisp_AreaReverseVRAM function reverses the specified area of VRAM.

```
void Bdisp_AreaReverseVRAM(
    int x1,            // x coordinate of the upper-left corner
    int y1,            // y coordinate of the upper-left corner
    int x2,            // x coordinate of the lower-right corner
    int y2             // y coordinate of the lower-right corner
);
```

Parameters

x1 (0 ~ 127)

This is the x coordinate of the upper-left corner.

y1 (0 ~ 63)

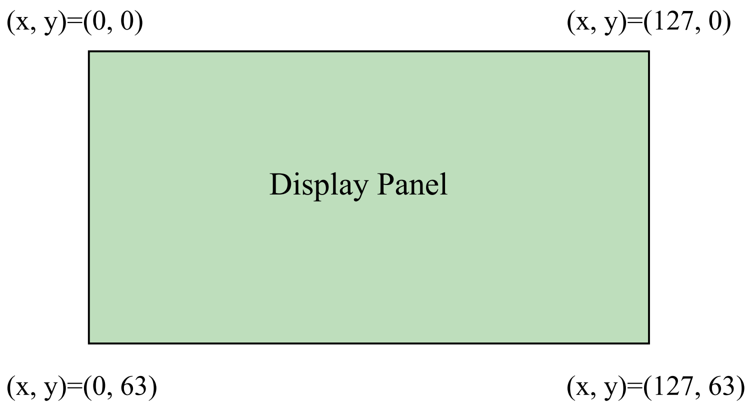
This is the y coordinate of the upper-left corner.

x2 (0 ~ 127)

This is the x coordinate of the lower-right corner.

y2 (0 ~ 63)

This is the y coordinate of the lower-right corner.



The coordinates of the two points specify the upper-left and lower-right corners of the rectangle. The sides of the rectangle extend from these two points and are parallel to the x-axis and the y-axis.

Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

Bdisp_GetDisp_DD/ Bdisp_GetDisp_VRAM

These functions get the bitmap data of all screens from VRAM or DD (Display Driver).

```
void Bdisp_GetDisp_DD(
void Bdisp_GetDisp_VRAM(
    unsigned char * pData        // pointer to the area where the bitmap is stored
);
```

Parameters

pData

This is a pointer to the area where the bitmap is stored. You should be securing the area of the size in 1024 bytes.

```
#define IM_VRAM_SIZE     1024        (Declared in dispbios.h)
```

Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

Bdisp_PutDisp_DD

The Bdisp_PutDisp_DD function transfers all bitmap data from VRAM to DD (Display Driver).

```
void Bdisp_PutDisp_DD(void);
```

Parameters

Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

The Bdisp_PutDisp_DD function is called in the GetKey function too.

Bdisp_PutDispArea_DD

The Bdisp_PutDispArea_DD function transfers the specified area from VRAM to DD (Display Driver).

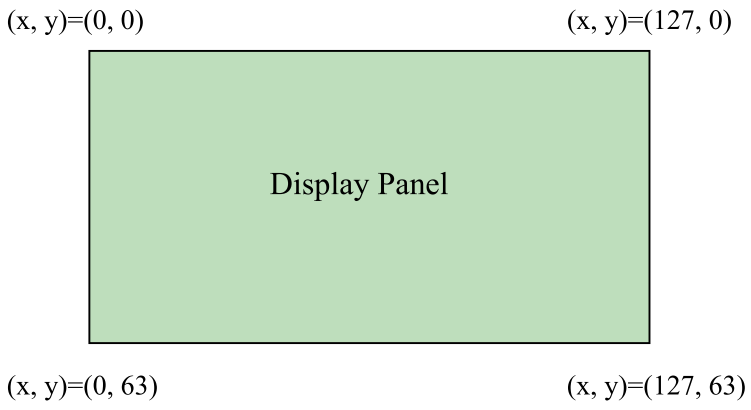
```
void Bdisp_AreaClr_DD(
void Bdisp_AreaClr_VRAM(
void Bdisp_AreaClr_DDVRAM(
    const DISPBOX *pArea     // pointer to structure data
);
```

Parameters

pArea

This is a pointer to the DISPBOX structure. The DISPBOX structure defines a rectangle. The structure specifies the coordinates of two points: the upper-left and lower-right corners of the rectangle. The sides of the rectangle extend from these two points and are parallel to the x-axis and the y-axis.

```
typedef struct tag_DISPBOX{              // Declared in dispbios.h
    int     left;       // 0~127
    int     top;        // 0~63
    int     right;      // 0~127
    int     bottom;     // 0~63
} DISPBOX;
```



Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

Bdisp_SetPoint_DD/ Bdisp_SetPoint_VRAM/ Bdisp_SetPoint_DDVRAM

These functions set or erase a dot at the specified position of VRAM and/or DD (Display Driver).

```
void Bdisp_SetPoint_DD(
void Bdisp_SetPoint_VRAM(
void Bdisp_SetPoint_DDVRAM(
    int x,                        // x coordinate
    int y,                        // y coordinate
    unsigned char point           // kind of dot
);
```

Parameters

x (0~127)

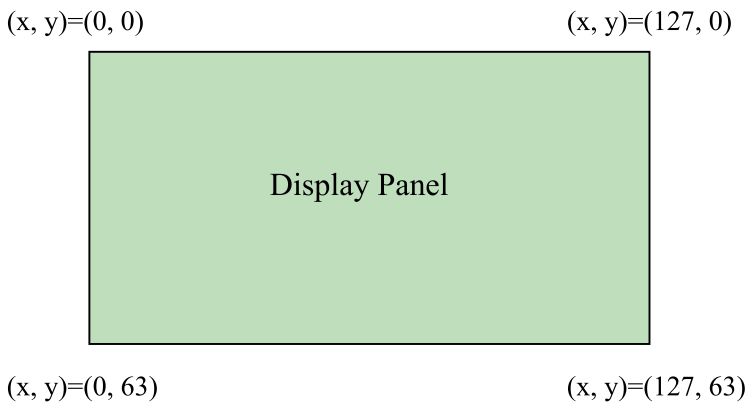
This is the x coordinate of the specified position.

y (0~63)

This is the y coordinate of the specified position.

point

If you set *point* to 1 then the dot is made black. If you set *point* to 0 then the dot is cleared.



Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

Bdisp_GetPoint_VRAM

The Bdisp_GetPoint_VRAM function checks a dot at the specified position of VRAM.

```
int Bdisp_GetPoint_VRAM(
    int x,              // x coordinate
    int y,              // y coordinate
);
```

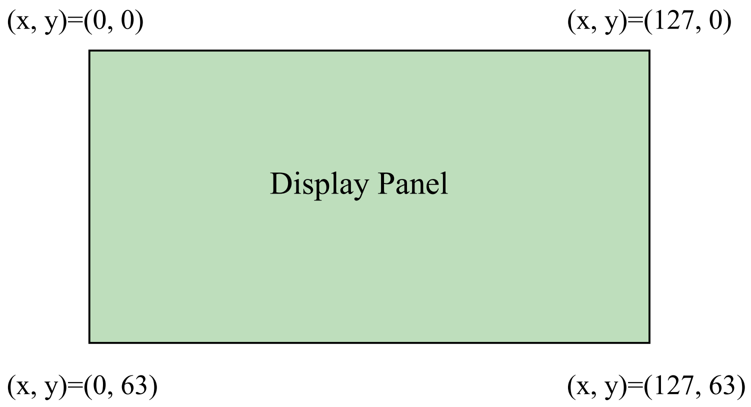
Parameters

x (0~127)

This is the x coordinate of the specified position.

y (0~63)

This is the y coordinate of the specified position.



Return Values

If the dot at the specified position is black then this function returns 1. If the dot at the specified position is white then this function returns 0.

QuickInfo

Header: Declared in fxlib.h.

Remarks

Bdisp_WriteGraph_DD/ Bdisp_WriteGraph_VRAM/ Bdisp_WriteGraph_DDVRAM

These functions copy a bitmap from a source rectangle to VRAM and/or DD (Display Driver).

```
void Bdisp_WriteGraph_DD(
void Bdisp_WriteGraph_VRAM(
void Bdisp_WriteGraph_DDVRAM(
    const DISPGRAPH *WriteGraph      // pointer to structure data
);
```

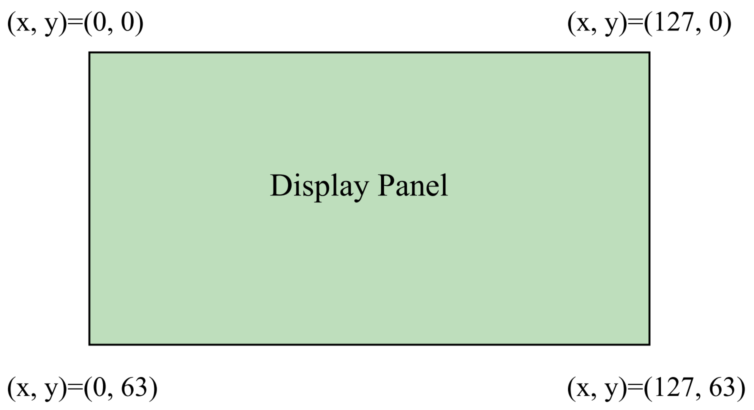
Parameters

WriteGraph

This is the pointer to the DISPGRAPH structure.

```
typedef struct tag_GRAPHDATA{          // Declared in dispbios.h.
    int                width;          // bitmap width
    int                height;         // bitmap height
    unsigned char      *pBitmap;       // pointer to bitmap data
} GRAPHDATA;

typedef struct tag_DISPGRAPH{                  // Declared in dispbios.h.
    int                        x;              // x coordinate of the upper-left corner
    int                        y;              // y coordinate of the upper-left corner
```



```
    GRAPHDATA              GraphData;          // pointer to structure data
    WRITEMODIFY            WriteModify;        // write modification graphic (declared in dispbios.h)
```

IMB_WRITEMODIFY_NORMAL	Normal
IMB_WRITEMODIFY_REVERSE	Reverse
IMB_WRITEMODIFY_MESH	Half-tone dot meshing

WRITEKIND WriteKind; // kind of writing (declared in dispbios.h)

IMB_WRITEKIND_OVER	Overwrite (fill)
IMB_WRITEKIND_OR	OR*[1]
IMB_WRITEKIND_AND	AND*[1]
IMB_WRITEKIND_XOR	XOR*[1]

*[1] When you use the Bdisp_WriteGraph_DDVRAM function, the function computes OR/AND/XOR of the bitmap data and VRAM data.

} DISPGRAPH;

Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

The bitmap data format is monochrome. The monochrome bitmap uses a one-bit, one- pixel format. Each scan is a multiple of 8 bits. If the corresponding bit in the bitmap is 1, the pixel is set to black. If the corresponding bit in the bitmap is zero, the pixel is set to white. Pixels are stored starting in the top left corner going from left to right and then row by row from the top the bottom.

Bdisp_ReadArea_DD / Bdisp_ReadArea_VRAM

These functions get the bitmap data of the specified area from VRAM or DD (Display Driver).

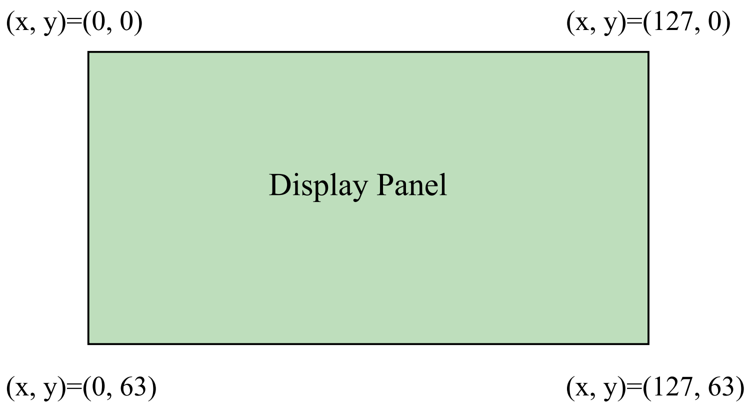
```
void Bdisp_ReadArea_DD (
void Bdisp_ReadArea_VRAM (
   const DISPBOX *ReadArea,          // pointer to structure data
   unsigned char *ReadData           // pointer to the area where the bitmap is stored
);
```

Parameters

ReadArea

This is a pointer to the DISPBOX structure. The DISPBOX structure defines a rectangle. The structure specifies the coordinates of two points: the upper-left and lower-right corners of the rectangle. The sides of the rectangle extend from these two points and are parallel to the x-axis and the y-axis.

```
typedef struct tag_DISPBOX{          // Declared in dispbios.h
   int     left;      // 0~127
   int     top;       // 0~63
   int     right;     // 0~127
   int     bottom;    // 0~63
} DISPBOX;
```



ReadData

This is a pointer to the area where the bitmap is stored.

Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

Bdisp_DrawLineVRAM

The Bdisp_DrawLineVRAM function draws a line to VRAM.

```
void Bdisp_DrawLineVRAM(
    int x1,              // x coordinate of the line's starting point
    int y1,              // y coordinate of the line's starting point
    int x2,              // x coordinate of the line's ending point
    int y2               // y coordinate of the line's ending point
);
```

Parameters

x1 (0 ~ 127)

This is the x coordinate of the line's starting point.

y1 (0 ~ 63)

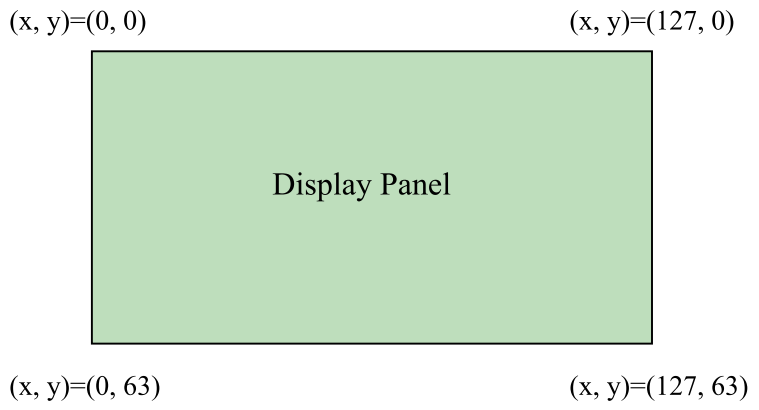
This is the y coordinate of the line's starting point.

x2 (0 ~ 127)

This is the x coordinate of the line's ending point.

y2 (0 ~ 63)

This is the y coordinate of the line's ending point.



Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

Bdisp_ClearLineVRAM

The Bdisp_ClearLineVRAM function draws a white line to VRAM.

```
void Bdisp_ClearLineVRAM(
    int x1,          // x coordinate of the line's starting point
    int y1,          // y coordinate of the line's starting point
    int x2,          // x coordinate of the line's ending point
    int y2           // y coordinate of the line's ending point
);
```

Parameters

x1　　(0 ~ 127)

This is the x coordinate of the line's starting point.

y1　　(0 ~ 63)

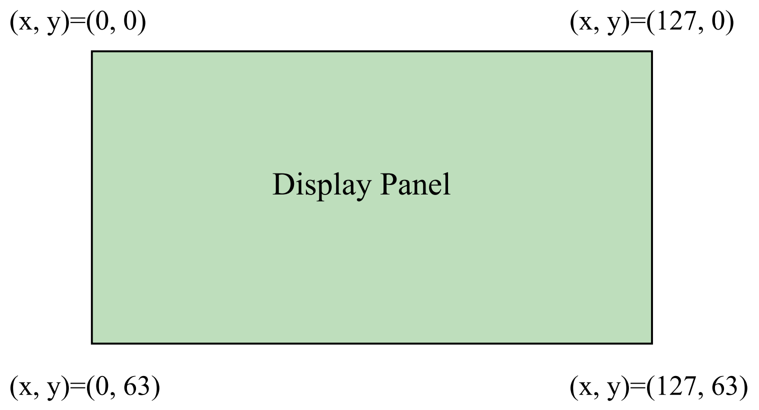
This is the y coordinate of the line's starting point.

x2　　(0 ~ 127)

This is the x coordinate of the line's ending point.

y2　　(0 ~ 63)

This is the y coordinate of the line's ending point.



Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

locate

The Locate function makes settings for the location of the display cursor.

```
void locate(
    int x,              // x position
    int y               // y position
);
```

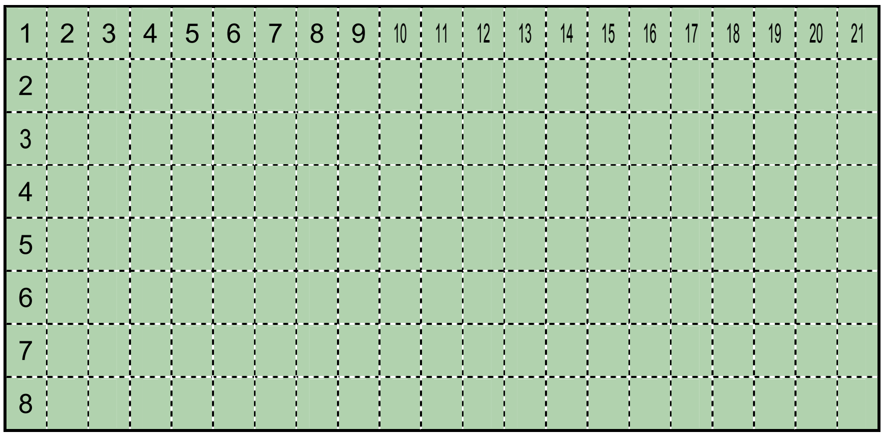
Parameters

x (1 ~ 21)

This is the x position of the display cursor.

y (1 ~ 8)

This is the y position of the display cursor.



Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

See Also

Print, PrintRev, PrintC, PrintRevC, PrintLine, PrintRLine

Print

The Print function displays a character string at the current position of the display cursor.

```
void Print(
    const unsigned char *str     // pointer to string
);
```

Parameters

str

This is the pointer to a character string. The character string must be null-terminated.

Return Values

QuickInfo

Header: Declared in fxlib.h.

Code set: fx-9860G Character Set.pdf

Remarks

After you call the Print function, the cursor position is moved to the right of the last character.

See Also

locate

PrintRev

The PrintRev function displays a character string in reversed color at the current position of the display cursor.

```
void PrintRev(
    const unsigned char *str     // pointer to string
);
```

Parameters

str

This is the pointer to a character string. The character string must be null-terminated.

Return Values

QuickInfo

Header: Declared in fxlib.h.

Code set: fx-9860G Character Set.pdf

Remarks

After you call the PrintRev function, the cursor position is moved to the right of the last character.

See Also

locate

PrintC

The PrintC function displays one character at the current position of the display cursor.

```
void PrintC(
    const unsigned char * c      // pointer to string
);
```

Parameters

c

This is the pointer to a character string. The character string must be null-terminated.

Return Values

QuickInfo

Header: Declared in fxlib.h.

Code set: fx-9860G Character Set.pdf

Remarks

After you call the PrintC function, the cursor position is moved to the right of the last character.

See Also

locate

PrintRevC

The PrintRevC function displays one character in reversed color at the current position of the display cursor.

```
void PrintRevC(
    const unsigned char *c       // pointer to string
);
```

Parameters

c

This is the pointer to a character string. The character string must be null-terminated.

Return Values

QuickInfo

Header: Declared in fxlib.h.

Code set: fx-9860G Character Set.pdf

Remarks

After you call the PrintRevC function, the cursor position is moved to the right of the last character.

See Also

locate

PrintLine

The PrintLine function displays a character string from the current position of the display cursor to the specified ending x position. If the character string ends before the ending position, the space up to the specified position is padded with blanks.

```
void PrintLine(
    const unsigned char *str,    // pointer to string
    int max                      // ending x position
);
```

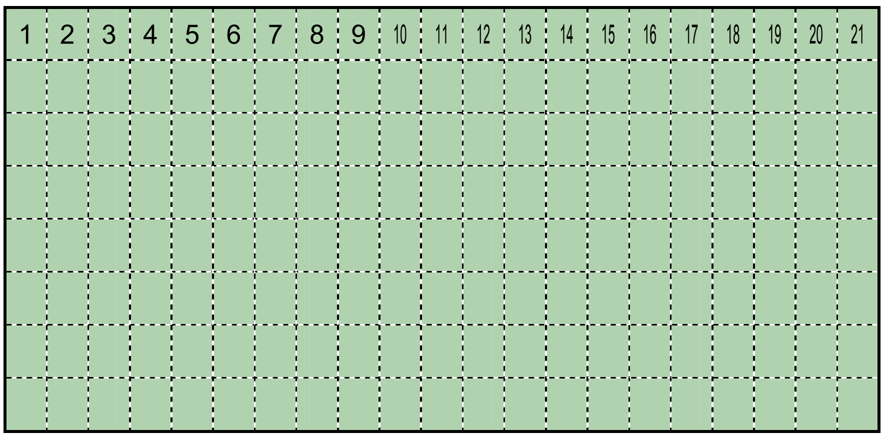
Parameters

str

This is the pointer to a character string. The character string must be null-terminated.

max (1 ~ 21)

This is the ending x position.



Return Values

QuickInfo

Header: Declared in fxlib.h.

Code set: fx-9860G Character Set.pdf

Remarks

After you call the PrintLine function, the cursor position is moved to the right of the last character.

See Also

locate

PrintRLine

The PrintLine function displays a character string in reversed color from the current position of the display cursor to the specified ending x position. If the character string ends before the ending position, the space up to the specified position is padded with blanks.

```
void PrintLine(
    const unsigned char *str,    // pointer to string
    int max                      // ending x position
);
```

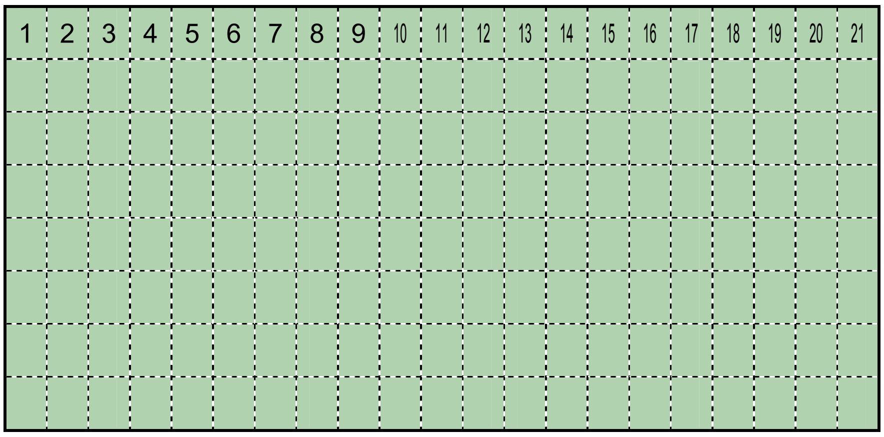
Parameters

str

This is the pointer to a character string. The character string must be null-terminated.

max (1 ~ 21)

This is the ending x position.



Return Values

QuickInfo

Header: Declared in fxlib.h.

Code set: fx-9860G Character Set.pdf

Remarks

After you call the PrintRLine function, the cursor position is moved to the right of the last character.

See Also

locate

PrintXY

The PrintXY function displays a character string at the specified position.

```
void PrintXY(
    int x;                        // x coordinate of the upper-left corner of the string
    int y;                        // y coordinate of the upper-left corner of the string
    const unsigned char *str,     // pointer to string
    int type                      // drawing type
);
```

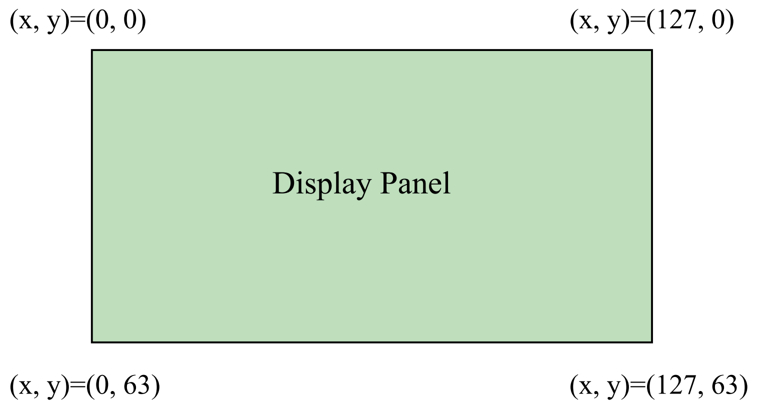
Parameters

x (0 ~ 127)

This is the x coordinate of the upper-left corner of the string

y (0 ~ 63)

This is the y coordinate of the upper-left corner of the string



str

This is the pointer to a character string. The character string must be null-terminated.

type

If you set the *type* to 0 then the character string is displayed in normal color. If you set the *type* to 1 then the character string is displayed in reverse color.

Return Values

QuickInfo

Header: Declared in fxlib.h.

Code set: fx-9860G Character Set.pdf

Remarks

PrintMini

The PrintMini function displays a small font character string at the specified position.

```
void PrintMini(
    int x;                      // x coordinate of the upper-left corner of the string
    int y;                      // y coordinate of the upper-left corner of the string
    const unsigned char *str,   // pointer to string
    int type                    // drawing type
);
```

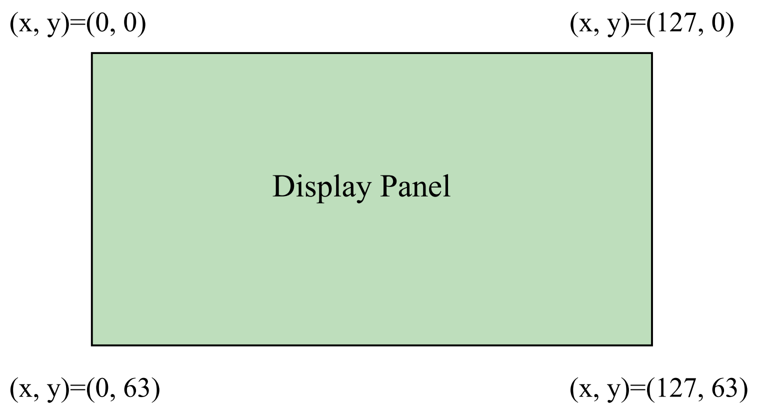
Parameters

x (0 ~ 127)

This is the x coordinate of the upper-left corner of the string

y (0 ~ 63)

This is the y coordinate of the upper-left corner of the string



str

This is the pointer to a character string. The character string must be null-terminated.

type

This is the drawing type. The following definitions are declared in dispbios.h.

MINI_OVER	Overwrite (fill)
MINI_OR	OR
MINI_REV	Reverse color
MINI_REVOR	OR in reverse color

Return Values

QuickInfo

Header: Declared in fxlib.h.

Code set: fx-9860G Character Set.pdf

Remarks

SaveDisp

The SaveDisp function saves a screen image in the system area.

```
void SaveDisp(
    unsigned char num           // number of the save area
);
```

Parameters

num

This is the number of the save area. The following definitions are declared in dispbios.h.

SAVEDISP_PAGE1
SAVEDISP_PAGE2
SAVEDISP_PAGE3

Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

The screen image is copied from VRAM. The image of the DD (Display Driver) is not copied.

See Also

RestoreDisp

RestoreDisp

The RestoreDisp function restores a screen image in the system area.

```
void RestoreDisp(
    unsigned char num           // number of the save area
);
```

Parameters

num

This is the number of the save area. The following definitions are declared in dispbios.h.

SAVEDISP_PAGE1
SAVEDISP_PAGE2
SAVEDISP_PAGE3

Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

Even if you call this function, the screen image is not displayed. You should call the Bdisp_PutDisp_DD function too.

See Also

SaveDisp

PopUpWin

The PopUpWin function displays a popup window of the specified size of lines. Also, the inside of the popup window is cleared.

```
void PopUpWin(
   int n                    // size of lines
);
```

Parameters

n   (1~5)

  This is the number of lines of the popup window.

Return Values

QuickInfo

Header: Declared in fxlib.h.

Remarks

To display the contents (string, etc.) of a window, use the Locate function or the Print function.

Bfile_OpenFile

The Bfile_OpenFile function opens an existing file.

```
int Bfile_OpenFile(
    const FONTCHARACTER *filename,          // pointer to the file name
    int mode                                // open mode
);
```

Parameters

filename

This is the pointer to a null-terminated string that names the file to be opened.

mode

The *mode* parameter specifies the action to open. The following definitions are declared in filebios.h.

_OPENMODE_READ	Opens the file for reading only.
_OPENMODE_READ_SHARE	Opens the file and denies read access to other processes.
_OPENMODE_WRITE	Opens the file for writing only.
_OPENMODE_READWRITE	Opens the file for reading and writing.
_OPENMODE_READWRITE_SHARE	Opens the file with exclusive mode, denying both read and write access to other processes.

Return Values

If the function succeeds, the return value specifies a file handle. It is greater than or equal to 0.

If the function fails, the return value is an error code. It is a negative value.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

Remarks

This function opens a file in Storage Memory or the SD card. To open the file in Main Memory, call the Bfile_OpenMainMemory function.

If you open a file in Storage Memory, the pathname to the file is the following string.

```
FONTCHARACTER PathName[]={'¥¥','¥¥','f','l','s','0','¥¥','f','i','l','e','n','a','m','e','.','e','x','t',0};
```

The root directory (¥¥fls0¥) means Storage Memory area. This name is fixed.

If you open a file in SD card, the pathname to the file is the following string.

```
FONTCHARACTER PathName[]={'¥¥','¥¥','c','r','d','0','¥¥','f','i','l','e','n','a','m','e','.','e','x','t',0};
```

The root directory (¥¥crd0¥) means SD card area. This name is fixed.

The capital letter and the small letter of the file name are identified.

To close the file after use, call the Bfile_CloseFile function. Only four files can be opened at the same time.

See Also

Bfile_ReadFile, Bfile_WriteFile, Bfile_SeekFile, Bfile_CloseFile, Bfile_CreateFile

Bfile_OpenMainMemory

The Bfile_OpenMainMemory function opens an existing file.

```
int Bfile_OpenMainMemory(
    const unsigned char *name               // pointer to the file name
);
```

Parameters

name

This is the pointer to a null-terminated string that names the file to be opened. The file name is limited to eight characters.

Return Values

If the function succeeds, the return value specifies a file handle. It is greater than or equal to 0.

If the function fails, the return value is an error code. It is a negative value.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

Remarks

This function opens a file in Main Memory. To open the file in Storage Memory or the SD card, call the Bfile_OpenFile function.

You do not need to specify the folder. You only have to set only the file name to this parameter.

Ex. unsigned char name[]={"Settings"};

The capital letter and the small letter of the file name are distinguished.

To close the file after use, call the Bfile_CloseFile function. Only four files can be opened at the same time.

See Also

Bfile_ReadFile, Bfile_WriteFile, Bfile_SeekFile, Bfile_CloseFile, Bfile_CreateMainMemory

Bfile_ReadFile

The Bfile_ReadFile function reads data from a file, starting at the position indicated by the file pointer. After the read operation has been completed, the file pointer is adjusted by the number of bytes actually read.

```
int Bfile_ReadFile(
    int HANDLE,     // handle of the file to read
    void *buf,      // pointer to buffer that reads data
    int size,       // number of bytes to read
    int readpos     // specifies the position to read.
);
```

Parameters

HANDLE

This is the handle of the file to read. *HANDLE* should be the handle opened by the Bfile_OpenFile or Bfile_OpenMainMemory function.

buf

This is the pointer to the buffer that receives the data read from the file.

size

This is the number of bytes to be read from the file.

readpos

This is the starting position to read. If the *readpos* parameter is -1, this function reads data from the position indicated by the file pointer. If the *readpos* parameter greater than or equal to 0, this function reads data from the position indicated by the *readpos* parameter.

Return Values

If the function succeeds, this function returns the number of bytes actually read. It is greater than or equal to 0.
If the function fails, the return value is an error code. It is a negative value.

QuickInfo

Error Code: Declared in filebios.h.
Header: Declared in fxlib.h.
Utility: Declared in endian.h.

Remarks

If you read the Windows file, the multi byte data is stored in Little Endian format. To use this data, you should convert the multi byte data into the Big Endian format.

See Also

Bfile_OpenFile, Bfile_OpenMainMemory, Bfile_SeekFile

Bfile_WriteFile

The Bfile_WriteFile function writes data to a file. The function starts writing data to the file at the position indicated by the file pointer. After the write operation has been completed, the file pointer is adjusted by the number of bytes actually written.

```
int Bfile_WriteFile(
    int HANDLE,              // handle of the file to write
    const void *buf,         // pointer to buffer that writes data
    int size                 // number of bytes to write
);
```

Parameters

HANDLE

This is the handle of the file to write. *HANDLE* should be the handle opened by the Bfile_OpenFile or Bfile_OpenMainMemory function.

buf

This is the pointer to the buffer containing the data to be written to the file.

size

This is the number of bytes to write to the file.

Return Values

If the function succeeds, this function returns the position indicated by the file pointer. It is greater than or equal to 0.

If the function fails, the return value is an error code. It is a negative value.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

Utility: Declared in endian.h.

Remarks

If you use a file that is written by this function on the PC, you should save the multi byte data in the Little Endian format.

See Also

Bfile_OpenFile, Bfile_OpenMainMemory, Bfile_SeekFile

Bfile_SeekFile

The Bfile_SeekFile function moves the file pointer of an open file.

```
int Bfile_SeekFile(
    int HANDLE,     // handle of the file
    int pos         // number of bytes to move file pointer
);
```

Parameters

HANDLE

This is the file handle whose file pointer will be moved *HANDLE* should be the handle opened by the Bfile_OpenFile or Bfile_OpenMainMemory function.

pos

This is the number of bytes to move file pointer.

Return Values

If the function succeeds, this function returns the number of bytes that can continuously be read. It is greater than or equal to 0.

If the function fails, the return value is an error code. It is a negative value.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

Remarks

The starting point for the file pointer move is the beginning of the file. The file pointer is at the head of the file immediately after the file was opened.

See Also

Bfile_OpenFile, Bfile_OpenMainMemory, Bfile_ReadFile, Bfile_WriteFile

Bfile_CloseFile

The Bfile_CloseFile function closes an open file handle.

```
int Bfile_CloseFile(
    int HANDLE      // handle of the file
);
```

Parameters

HANDLE

This is the handle of the file to close. *HANDLE* should be the handle opened by the Bfile_OpenFile or Bfile_OpenMainMemory function.

Return Values

If the function succeeds, this function returns 0.

If the function fails, the return value is an error code. It is a negative value.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

Remarks

Only four files can be opened at the same time.

See Also

Bfile_OpenFile, Bfile_OpenMainMemory

Bfile_GetMediaFree

The Bfile_GetMediaFree function retrieves the size of the free space, in bytes, of the specified device.

```
int Bfile_GetMediaFree(
    enum DEVICE_TYPE devicetype,    // type of device
    int *freebytes                  // pointer to the size of the free space
);
```

Parameters

devicetype

This is the type of device. The following definitions are declared in filebios.h.

DEVICE_MAIN_MEMORY	Main Memory
DEVICE_STORAGE	Storage Memory
DEVICE_SD_CARD	SD card (only fx-9860G SD model)

freebytes

This is the pointer to the size of the free space.

Return Values

If the function succeeds, this function returns 0.

If the function fails, the return value is an error code. It is a negative value.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

Remarks

See Also

Bfile_CreateFile, Bfile_CreateMainMemory, Bfile_WriteFile

Bfile_GetFileSize

The Bfile_GetFileSize function retrieves the size, in bytes, of the specified file.

```
int Bfile_GetFileSize(
    int HANDLE
);
```

Parameters

HANDLE

This is the handle of the file whose size will be retrieved. *HANDLE* should be the handle opened by the Bfile_OpenFile or Bfile_OpenMainMemory function.

Return Values

If the function succeeds, the return value is the file size.

If the function fails, the return value is an error code. It is a negative value.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

Remarks

See Also

Bfile_OpenFile, Bfile_OpenMainMemory, Bfile_WriteFile

Bfile_CreateFile

The Bfile_CreateFile function creates the file.

```
int Bfile_CreateFile(
    const FONTCHARACTER *filename,          // pointer to the file name
    int size                                // size of the file to be created
);
```

Parameters

filename

This is the pointer to a null-terminated string that names the file to be created.

size

This is the size of the file to be created. This parameter is only used when you create a file in Storage Memory.

Return Values

If the function succeeds, the return value is 0.

If the function fails, the return value is an error code. It is a negative value.

Remarks

This function creates a file in Storage Memory or the SD card. To create the file in Main Memory, call the Bfile_CreateMainMemory function.

If you write data into a created file, you should call the Bfile_OpenFile function and Bfile_WriteFile function too.

If you create a file in Storage Memory, the pathname to the file is the following string.

```
    FONTCHARACTER PathName[]={'¥¥','¥¥','f','l','s','0','¥¥','f','i','l','e','n','a','m','e','.','e','x','t',0};
```

The root directory (¥¥fls0¥) means Storage Memory area. This name is fixed.

If you create a file in SD card, the pathname to the file is the following string.

```
    FONTCHARACTER PathName[]={'¥¥','¥¥','c','r','d','0','¥¥','f','i','l','e','n','a','m','e','.','e','x','t',0};
```

The root directory (¥¥crd0¥) means SD card area. This name is fixed.

The capital letter and the small letter of the file name are identified.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

See Also

Bfile_OpenFile, Bfile_WriteFile

Bfile_CreateDirectory

The Bfile_CreateDirectory function creates a new directory.

```
int Bfile_CreateDirectory(
    const FONTCHARACTER *pathname                    // pointer to directory path string
);
```

Parameters

pathname

This is the pointer to a null-terminated string that specifies the path of the directory to be created.

Return Values

If the function succeeds, the return value is 0.

If the function fails, the return value is an error code. It is a negative value.

Remarks

This function creates a directory in Storage Memory or the SD card. You cannot create a directory in Main Memory.

If you create a directory in Storage Memory, the pathname to the directory is the following string.

    FONTCHARACTER PathName[]={'¥¥','¥¥','f','l','s','0','¥¥','d','i','r','n','a','m','e',0};

The root directory (¥¥fls0¥) means Storage Memory area. This name is fixed.

If you create a directory in SD card, the pathname to the directory is the following string.

    FONTCHARACTER PathName[]={'¥¥','¥¥','c','r','d','0','¥¥','d','i','r','n','a','m','e',0};

The root directory (¥¥crd0¥) means SD card area. This name is fixed.

The capital letter and the small letter of the file name are identified.

The directory can be created under only the route directory.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

See Also

Bfile_CreateFile

Bfile_CreateMainMemory

The Bfile_CreateMainMemory function creates the file.

```
int Bfile_CreateMainMemory(
    const unsigned char *name
);
```

Parameters

name

This is the pointer to a null-terminated string that names the file to be created. The file name is limited to eight characters.

Return Values

If the function succeeds, the return value is 0.

If the function fails, the return value is an error code. It is a negative value.

Remarks

This function creates a file in Main Memory. To create the file in Storage Memory or the SD card, call the Bfile_CreateFile function.

You do not need to specify the folder. You only have to set only the file name to this parameter.

Ex. unsigned char name[]={"Settings"};

The capital letter and the small letter of the file name are distinguished.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

See Also

Bfile_OpenMainMemory, Bfile_WriteFile

Bfile_RenameMainMemory

The Bfile_RenameMainMemory function renames the specified file.

```
int Bfile_RenameMainMemory(
    const unsigned char *oldname,        // the file to rename
    const unsigned char *newname         // the new name of the file
);
```

Parameters

oldname

This is the pointer to a null-terminated string that specifies the file to rename.

newname

This is the pointer to a null-terminated string that specifies the new name of the file. The file name is limited to eight characters.

Return Values

If the function succeeds, the return value is 0.

If the function fails, the return value is an error code. It is a negative value.

Remarks

This function renames a file in Main Memory. To rename the file in Storage Memory or the SD card, call the Bfile_RenameFile function.

You do not need to specify the folder. You only have to set only the file name to this parameter.

Ex. unsigned char name[]={“Settings”};

The capital letter and the small letter of the file name are distinguished.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

See Also

Bfile_DeleteFile

The Bfile_DeleteFile function deletes an existing file.

```
int Bfile_DeleteFile(
    const FONTCHARACTER *filename              // pointer to the file name
);
```

Parameters

filename

This is the pointer to a null-terminated string that specifies the file to delete.

Return Values

If the function succeeds, the return value is 0.

If the function fails, the return value is an error code. It is a negative value.

Remarks

This function deletes a file in Storage Memory or the SD card. To delete the file in Main Memory, call the Bfile_DeleteMainMemory function.

If you delete a directory in Storage Memory, the pathname to the directory is the following string.

    FONTCHARACTER PathName[]={'¥¥','¥¥','f','l','s','0','¥¥','f','i','l','e','n','a','m','e','.','e','x','t',0};

The root directory (¥¥fls0¥) means Storage Memory area. This name is fixed.

If you delete a directory in SD card, the pathname to the directory is the following string.

    FONTCHARACTER PathName[]={'¥¥','¥¥','c','r','d','0','¥¥','f','i','l','e','n','a','m','e','.','e','x','t',0};

The root directory (¥¥crd0¥) means SD card area. This name is fixed.

The capital letter and the small letter of the file name are identified.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

See Also

Bfile_DeleteDirectory

The Bfile_DeleteDirectory function deletes an existing directory.

```
int Bfile_DeleteDirectory(
    const FONTCHARACTER *pathname            // pointer to the directory name
);
```

Parameters

pathname

This is the pointer to a null-terminated string that specifies the directory to delete.

Return Values

If the function succeeds, the return value is 0.

If the function fails, the return value is an error code. It is a negative value.

Remarks

This function deletes a directory in Storage Memory. You cannot delete a directory in Main Memory and the SD card.

If you delete a directory in Storage Memory, the pathname to the file is the following string.

```
    FONTCHARACTER PathName[]={'¥¥','¥¥','f','l','s','0','¥¥','d','i','r','n','a','m','e',0};
```

The root directory (¥¥fls0¥) means Storage Memory area. This name is fixed.

The capital letter and the small letter of the file name are identified.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

See Also

Bfile_DeleteMainMemory

The Bfile_DeleteMainMemory function deletes an existing file.

```
int Bfile_DeleteMainMemory(
    const unsigned char *name           // pointer to the file name
);
```

Parameters

name

This is the pointer to a null-terminated string that specifies the file to delete.

Return Values

If the function succeeds, the return value is 0.

If the function fails, the return value is an error code. It is a negative value.

Remarks

This function deletes a file in Main Memory. To delete the file in Storage Memory or the SD card, call the Bfile_DeleteFile function.

You do not need to specify the folder. You only have to set only the file name to this parameter.

    Ex. unsigned char name[]={“Settings”};

The capital letter and the small letter of the file name are distinguished.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

See Also

Bfile_FindFirst

The Bfile_FindFirst function searches a directory for a file whose name matches the specified filename. The Bfile_FindFirst examines subdirectory names as well as filenames.

```
int Bfile_FindFirst(
    const FONTCHARACTER *pathname,          // pointer to pathname
    int *FindHandle,                        // pointer to handle
    FONTCHARACTER *foundfile,               // pointer to found filename
    FILE_INFO *fileinfo                     // pointer to structure
);
```

Parameters

pathname

This is the pointer to a null-terminated string that specifies a valid directory or path and filename, which can contain wildcard characters "*".

FindHandle

If this function success, the return value is a search handle used in a subsequent call to Bfile_FindNext or Bfile_FindClose.

foundfile

This is the pointer to the buffer that receives the found file or subdirectory

fileinfo

This is the pointer to the FILE_INFO structure that receives information about the found file or subdirectory.

```
typedef struct tag_FILE_INFO                // declared in filebios.h
{
        unsigned short        id;           // file index
        unsigned short        type;         // file type (declared in filebios.h.)
```

DT_DIRECTORY	Directory
DT_FILE	File
DT_ADDIN_APP	Add-In application
DT_EACT	eActivity
DT_LANGUAGE	Language
DT_BITMAP	Bitmap
DT_MAINMEM	Main Memory data
DT_TEMP	Temporary data
DT_DOT	"." (Current directory)
DT_DOTDOT	".." (Parent directory)
DT_VOLUME	Volume label

```
    unsigned long       fsize;          // file size
    unsigned long       dsize;          // data size (except file header)
    unsigned int        property;       // the file has not been completed, except when property is 0.
    unsigned long       address;        // top address of data (Don't directly access this address.)
} FILE_INFO;
```

Return Values

If the function succeeds, the return value is 0. When a specified file is not found, this function returns IML_FILEERR_ENUMRATEEND (declared in filebios.h).

If the function fails, the return value is an error code. It is a negative value.

Remarks

This function finds files and subdirectories in Storage Memory or the SD card. You cannot find files and subdirectories in Main Memory.

If you find the Bitmap files in Storage Memory, the pathname is the following string.

```
FONTCHARACTER PathName[]={'¥¥','¥¥','f','l','s','0','¥¥','*','.','b','m','p',0};
```

The root directory (¥¥fls0¥) means Storage Memory area. This name is fixed.

If you find the Bitmap files in SD card, the pathname is the following string.

```
FONTCHARACTER PathName[]={'¥¥','¥¥','c','r','d','0','¥¥','*','.','b','m','p',0};
```

The root directory (¥¥crd0¥) means SD card area. This name is fixed.

The capital letter and the small letter of the file name are identified.

When the search handle is no longer needed, close it by using the Bfile_FindClose function. Only four search handles can be opened at the same time.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

See Also

Bfile_FindNext, Bfile_FindClose

Bfile_FindNext

The Bfile_FindNext function continues a file search from a previous call to the Bfile_FindFirst function.

```
int Bfile_FindNext(
    int FindHandle,                 // handle to search
    FONTCHARACTER *foundfile,       // pointer to found filename
    FILE_INFO *fileinfo             // pointer to structure
);
```

Parameters

FindHandle

This is a search handle returned by a previous call to the Bfile_FindFirst function.

foundfile

This is the pointer to the buffer that receives the found file or subdirectory

fileinfo

This is the pointer to the FILE_INFO structure that receives information about the found file or subdirectory.

```
typedef struct tag_FILE_INFO                    // declared in filebios.h
{
        unsigned short      id;                 // file index
        unsigned short      type;               // file type (declared in filebios.h.)
```

DT_DIRECTORY	Directory
DT_FILE	File
DT_ADDIN_APP	Add-In application
DT_EACT	eActivity
DT_LANGUAGE	Language
DT_BITMAP	Bitmap
DT_MAINMEM	Main Memory data
DT_TEMP	Temporary data
DT_DOT	"." (Current directory)
DT_DOTDOT	".." (Parent directory)
DT_VOLUME	Volume label

```
        unsigned long       fsize;              // file size
        unsigned long       dsize;              // data size (except file header)
        unsigned int        property;           // the file has not been completed, except when property is 0.
        unsigned long       address;            // top address of data (Don't directly access this address.)
} FILE_INFO;
```

Return Values

If the function succeeds, the return value is 0. When a specified file is not found, this function returns IML_FILEERR_ENUMRATEEND (declared in filebios.h).

If the function fails, the return value is an error code. It is a negative value.

Remarks

This function finds files and subdirectories in Storage Memory or the SD card. You cannot find files and subdirectories in Main Memory.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

See Also

Bfile_FindFirst, Bfile_FindClose

Bfile_FindClose

The Bfile_FindClose function closes the specified search handle. The Bfile_FindFirst and Bfile_FindNext functions use the search handle to locate files with names that match a given name.

```
int Bfile_FindClose(
    int FindHandle                  // file search handle
);
```

Parameters

FindHandle

This is a search handle. This handle must have been previously opened by the Bfile_FindFirst function.

Return Values

If the function succeeds, the return value is 0.

If the function fails, the return value is an error code. It is a negative value.

Remarks

Only four search handles can be opened at the same time.

QuickInfo

Error Code: Declared in filebios.h.

Header: Declared in fxlib.h.

See Also

Bfile_FindFirst, Bfile_FindNext

Bkey_Set_RepeatTime

The Bkey_Set_RepeatTime is sets the repetition interval of the key.

```
void Bkey_Set_RepeatTime(
    long FirstCount,          // repeat time of first repeat
    long NextCount            // repeat time of second repeat
);
```

Parameters

FirstCount

This is time from pushing the key to the input of the first repetition key. This parameter is the time in milliseconds divided by 25.

NextCount

This is the interval time of the repetition key. This parameter is the time in milliseconds that was divided by 25.

Return Values

Remarks

The cursor key is the only key that can be set for repetition. Other keys will not repeat even if they are continuously pressed.

QuickInfo

Header: Declared in fxlib.h.

See Also

Bkey_Get_RepeatTime, Bkey_Set_RepeatTime_Default, GetKey, GetKeyWait

Bkey_Get_RepeatTime

The Bkey_Get_RepeatTime is gets the repetition interval of the key.

```
void Bkey_Get_RepeatTime(
    long *FirstCount,           // pointer to repeat time of first repeat
    long *NextCount             // pointer to repeat time of second repeat
);
```

Parameters

FirstCount

This is the pointer to time from pushing the key to the input of the first repetition key. This parameter is the time in milliseconds divided by 25.

NextCount

This is the pointer to the interval time of the repetition key. This parameter is the time in milliseconds divided by 25.

Return Values

Remarks

The cursor key is the only key that can be set for repetition. Other keys will not repeat even if they are continuously pressed.

QuickInfo

Header: Declared in fxlib.h.

See Also

Bkey_Set_RepeatTime, Bkey_Set_RepeatTime_Default, GetKey, GetKeyWait

Bkey_Set_RepeatTime_Default

The Bkey_Set_RepeatTime_Default is sets the default key repetition interval.

```
void Bkey_Set_RepeatTime_Default(void);
```

Parameters

Return Values

Remarks

This function sets the time from pushing the key to starting key repeats to 500 milliseconds, and sets the interval time of repetition key to 125 milliseconds.

The cursor key is the only key that can be set for repetition. Other keys will not repeat even if they are continuously pressed.

QuickInfo

Header: Declared in fxlib.h.

See Also

Bkey_Set_RepeatTime, Bkey_Get_RepeatTime, GetKey, GetKeyWait

GetKeyWait

The GetKeyWait function performs a key wait and returns a value indicating the pressed key.

```
int GetKeyWait(
    int sel,                    // type of waiting
    int time,                   // time out period
    int menu,                   // operation of menu key
    unsigned int *keycode       // pointer to key code
);
```

Parameters

sel

This is the type of waiting for key. The following definitions are declared in keybios.h.

KEYWAIT_HALTON_TIMEROFF	If there are no characters in the key buffer, this function waits until a character arrives and then returns immediately.
KEYWAIT_HALTOFF_TIMEROFF	If there are no characters in the key buffer, this function returns immediately.
KEYWAIT_HALTON_TIMERON	If no character arrives within the time specified by the *time* parameter, this function times out.

time

This is the time out period in seconds. This parameter is used only when the first parameter is KEYWAIT_HALTON_TIMERON.

menu

If you set 0 to the *menu*, the menu key performs Main-Menu.

If you set 1 to the *menu*, the menu key returns the key code.

keycode

If the function succeeds, the function returns key code.

Return Values

The function will return the following values. The following definitions are declared in keybios.h.

KEYREP_NOEVENT	Because there are no characters in the key buffer, this function returned immediately.
KEYREP_KEYEVENT	*Keycode* will be set.
KEYREP_TIMEREVENT	This function returned because *time* seconds passed.

Remarks

The Auto Power Off event is also handled by this function. If an Auto Power Off event occurs the application will not be notified. Once the calculator is turned back on control is returned to the GetKeyWait function.

QuickInfo

Key code: fx-9860G Key Code List.pdf

Header: Declared in fxlib.h.

See Also

GetKey, Bkey_Set_RepeatTime, Bkey_Get_RepeatTime, Bkey_Set_RepeatTime_Default

GetKey

The GetKey function performs a key wait and returns a value indicating the pressed key.

```
int GetKey(
    unsigned int *keycode        // pointer to key code
);
```

Parameters

keycode

The function returns key code into the *keycode*.

Return Values

If a character key was pressed, the return value is 1.

If a control key was pressed, the return value is 0.

Remarks

If there are no characters in the key buffer, this function waits until a character arrives and then returns immediately.

Even if the menu key pressed, this function does not return menu key code. The system processes the menu key event.

Off event, Auto Power Off event, and contrast adjustment are also handled by this function. If an off or Auto Power Off event occurs the application will not be notified. Once the calculator is turned back on control is returned to the GetKey function.

QuickInfo

Key code: fx-9860G Key Code List.pdf

Header: Declared in fxlib.h.

See Also

GetKeyWait, Bkey_Set_RepeatTime, Bkey_Get_RepeatTime, Bkey_Set_RepeatTime_Default

IsKeyDown

The IsKeyDown function checks whether the specified key is pressed.

```
int IsKeyDown(
    int keycode         // key code
);
```

Parameters

keycode

This is a key code of the key that you check.

Return Values

If the specified key is pressed, the return value is 1.

If the specified key is not pressed, the return value is 0.

Remarks

When you wait events only by this function, the calculator never turns off. Only use this function when you need to wait for key input to continue.

QuickInfo

Key code: fx-9860G Key Code List.pdf

Header: Declared in fxlib.h.

See Also

IsKeyUp

IsKeyUp

The IsKeyUp function checks whether the specified key was completely released.

```
int IsKeyUp(
    int keycode        // key code
);
```

Parameters

keycode

This is a key code of the key that you check.

Return Values

If the specified key was completely released, the return value is 1.

If the specified key was completely released, the return value is 0.

Remarks

QuickInfo

Key code: fx-9860G Key Code List.pdf

Header: Declared in fxlib.h.

See Also

IsKeyDown

SetTimer

The SetTimer function creates a timer with the specified time-out value.

```
int SetTimer(
    int ID,                   // timer identifier
    int elapse,               // time-out value
    void (*hander)(void)      // address of timer procedure
);
```

Parameters

ID

This parameter specifies a timer identifier. The following defines are declared in timer.h.

ID_USER_TIMER1
ID_USER_TIMER2
ID_USER_TIMER3
ID_USER_TIMER4
ID_USER_TIMER5

elapse

This parameter specifies the time-out value, in milliseconds. It is set at 25ms intervals.

hander

This is the pointer to the function to be notified when the time-out value elapses.

Return Values

If the function succeeds, the return value is a timer identifier. This is the same as the first parameter.

If the function fails, the return value is a negative value. In this case, the specified timer in use or parameter is illegal.

Remarks

Do not do time-consuming processes in the timer handler.

QuickInfo

Header: Declared in fxlib.h.

See Also

KillTimer

KillTimer

The KillTimer function destroys the specified timer.

```
int KillTimer(
    int ID
);
```

Parameters

ID

This is specifies the timer to be destroyed. This parameter must be the timer identifier returned by SetTimer.

Return Values

If the function succeeds, the return value is a timer identifier. This is same as first parameter.

If the function fails, the return value is a negative value. In this case, parameter is illegal.

Remarks

QuickInfo

Header: Declared in fxlib.h.

See Also

SetTimer

Sleep

The Sleep function suspends the execution of the current thread for a specified interval.

```
void Sleep(
    int millisecond                    // sleep time in milliseconds
);
```

Parameters

millisecond

This is specifies the time, in milliseconds, for which to suspend execution.

Return Values

Remarks

QuickInfo

Header: Declared in fxlib.h.

See Also

SetQuitHandler

The SetQuitHandler function registers the handler that is called when the application terminates.

```
void SetQuitHandler(
   void (*callback)(void)        // address of handler
);
```

Parameters

callback

This is the pointer to the function to be called when the application is closing.

Return Values

Remarks

QuickInfo

Header: Declared in fxlib.h.

See Also

CASIO®

CASIO COMPUTER CO., LTD.

6-2, Hon-machi 1-chome
Shibuya-ku, Tokyo 151-8543, Japan

SA0701-A