

Driver module API for acquiring Printer Status (FTPCtrl.dll) Specifications

FCL COMPONENTS LIMITED
V2.00

The enclosed “FTPCtrl.DLL” and “FTPCtrl_x64.DLL” provides the exported functions to acquire the status of the USB thermal printer.

By calling the exported functions, “FTPCtrl.DLL” sends the vendor request to printer and notifies response data, the applications can get the printer status easily.

If exported function is called from x64 application, please use “FTPCtrl_x64.DLL”. And if exported function is called from x86 application, please use “FTPCtrl.DLL”.

This dll supports only printer class.

Trademark

- Microsoft®, Windows®, Visual Studio®, Visual C++®, Visual Basic® and ActiveX® are registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.
- The proper name of Windows is Microsoft Windows operating system.
- All other product and brand names are trademarks and/or registered trademarks of their respective companies.

Notes

- The contents in this manual are subject to change without notice.
- Any reproduction of this manual in part or in whole is prohibited.
- Our company is not liable for any damages resulting from the use of the information contained in this manual.
- The product names and service names are trademarks or registered trademarks of their respective owners.

– Exported DLL functions –

Exported DLL function	Effect
ULONG __cdecl FcITP_Search_USB	Search USB port driver and acquire the port driver handle.
ULONG __cdecl FcITP_GetVendorCommand	Send Vendor Commands to printer, and get response data from printer.
ULONG __cdecl FcITP_Search_USBEx	Search USB port driver related to printer driver name and acquire the port driver handle.
ULONG __cdecl FcITP_GetVendorCommandEx	Send Vendor Commands to printer related to printer driver name and get response data from printer.
ULONG __stdcall FcITP_Search_USB_B	Search USB port driver, and acquire the port driver handle.
ULONG __stdcall FcITP_GetVendorCommand_B	Send Vendor Commands to printer, and get response data from printer.
ULONG __stdcall FcITP_Search_USBEx_B	Search USB port driver related to printer driver name and acquire the port driver handle.
ULONG __stdcall FcITP_GetVendorCommandEx_B	Send Vendor Commands to printer related to printer driver name and get response data from printer.

– Specifications of exported DLL functions –

ULONG __cdecl FcITP_Search_USB (void)

Search USB port driver, and acquire the port driver handle.

Parameters

Nothing

Return Value

= 1 : Success

≠ 1 : Failure

Remarks

The port driver handle that acquired by calling FcITP_Search_USB function is saved in DLL until the FreeLibrary function is executed. It is necessary to call FcITP_Search_USB function once and acquire the handle before FcITP_GetVendorCommand function is executed.

If FcITP_Search_USB function fails, there is a possibility that printer driver is not normally installed.

**ULONG _cdecl FcITP_GetVendorCommand (PVENDOR_COMMAND *lpVendorCmd*,
PRECEIVE_DATA *lpRcvData*)**

Send Vendor Command, and acquire the response data from printer.

Parameters

lpVendorCmd

Pointer to the VENDOR_COMMAND structure.

lpRcvData

Pointer to the RECEIVE_DATA structure.

Return Value

= 1 : Success
 ≠ 1 : Failure

Remarks

FcITP_GetVendorCommand function send vendor request that is specified bRequest member of lpVendorCmd, and store response data to lpRcvData.

```
// VENDOR_COMMAND structure
typedef struct tagVENDOR_COMMAND
{
    USHORT    unitLength;           // VENDOR_COMMAND structure size, in bytes
    UCHAR     bRequest;             // specifies vendor request
    UCHAR     wValueH;              // high-order byte of wValue
    UCHAR     wValueL;              // low-order byte of wValue
    UCHAR     wIndexH;              // high-order byte of wIndex
    UCHAR     wIndexL;              // low-order byte of wIndex
    UCHAR     wLengthH;             // high-order byte of wLength
    UCHAR     wLengthL;             // low-order byte of wLength
} VENDOR_COMMAND, *PVENDOR_COMMAND;

// RECEIVE_DATA structure
typedef struct tagRECEIVE_DATA
{
    USHORT    unitLength;           // RECEIVE_DATA structure size, in bytes
    BOOLEAN   DataValid;            // data valid (TRUE) / data invalid(FALSE)
    ULONG     DataLength;           // response data size, in bytes
    UCHAR     Data[256];            // buffer to storage response data
} RECEIVE_DATA, *PRECEIVE_DATA;
```

For example, get the printer status, specifies the value for the members of lpVendorCmd, as follows

```
bRequest = 1;
wValueH = 0; wValueL = 0;
wIndexH = 0; wIndexL = 0;
wLengthH = 0; wLengthL = 4; //The length of data depends on the printer.
```

if FcITP_GetVendorCommand function fails, return value is not 1, and DataValid becomes FALSE.

- * For more information about the kind of vendor request, specifies the value for the members of lpVendorCmd and response data, see printer product specification.

ULONG _cdecl FclTP_Search_USBEx (LPSTR *lpPrinterDriverName*)

Search USB port driver related to printer driver name, and acquire the port driver handle.

Parameters

lpPrinterDriverName

Specify the printer name allocated when the printer driver is installed.

Return Value

= 1 : Success

≠ 1 : Failure

Remarks

FclTP_Search_USBEx function searches the USB port driver related to printer driver name and acquire the port driver handle.

The port handle is saved with the printer driver name specified with lpPrinterDriverName in DLL.

In addition, when another printer driver name is specified, and the USB port driver searches and the port handle acquisition succeed, it saves it besides the former data. Therefore, two or more printer driver name and handle can be saved in DLL.

The port driver handle that acquired by calling FclTP_Search_USB function is saved in DLL until the FreeLibrary function is executed.

It is necessary to call FclTP_Search_USB function once and acquire the handle before

FclTP_GetVendorCommand function is executed.

If FclTP_Search_USB function fails, there is a possibility that printer driver is not normally installed or the mistake is found in the printer driver name specified with lpPrinterDriverName.

ULONG _cdecl FcITP_GetVendorCommandEx (LPSTR *lpPrinterDriverName*,
PVENDOR_COMMAND *lpVendorCmd*,
PRECEIVE_DATA *lpRcvData*)

Send the Vendor Command to the printer related to printer driver name, and acquire the response data from printer.

Parameters

lpPrinterDriverName

Specify the printer name allocated when the printer driver is installed.

lpVendorCmd

Pointer to the VENDOR_COMMAND structure.

lpRcvData

Pointer to the RECEIVE_DATA structure.

Return Value

= 1 : Success
≠ 1 : Failure

Remarks

FcITP_GetVendorCommand function send the vendor request that is specified bRequest member of lpVendorCmd to the printer related to printer driver name, and store response data to lpRcvData.

When this function is executed, it is necessary to execute FcITP_Search_USBEx function, and to acquire the handle of USB port driver.

When only the FcITP_Search_USB function is executed, this function returns the failure because it cannot judge the USB printer related to the printer driver name.

The setting of the structure of lpVendorCmd and lpRcvData and the execution method are the same as the FcITP_GetVendorCommand function. Please refer to page 3 and 9 of this document.

ULONG _stdcall FcITP_Search_USB_B (void)

Search USB port driver, and acquire the port driver handle.

Parameters

Nothing

Return Value

= 1 : Success

≠ 1 : Failure

Remarks

The execution method are the same as the FcITP_SearchUSB function. Please refer to page 2 of this document.

**ULONG _stdcall FcITP_GetVendorCommand_B (PVENDOR_COMMAND *IpVendorCmd*,
PRECEIVE_DATA *IpRcvData*)**

Send Vendor Command, and acquire the response data from printer.

Parameters

IpVendorCmd

Pointer to the VENDOR_COMMAND structure.

IpRcvData

Pointer to the RECEIVE_DATA structure.

Return Value

= 1 : Success

≠ 1 : Failure

Remarks

The execution method are the same as the FcITP_GetVendorCommand function. Please refer to page 3 of this document.

ULONG __stdcall FcITP_Search_USBEx_B (LPSTR *lpPrinterDriverName*)

Search USB port driver related to printer driver name, and acquire the port driver handle.

Parameters

lpPrinterDriverName

Specify the printer name allocated when the printer driver is installed.

Return Value

= 1 : Success

≠ 1 : Failure

Remarks

The execution method are the same as the FcITP_SearchUSBEx function. Please refer to page 4 of this document.

**ULONG __stdcall FcITP_GetVendorCommandEx_B (LPSTR *lpPrinterDriverName*,
PVENDOR_COMMAND *lpVendorCmd*,
PRECEIVE_DATA *lpRcvData*)**

Send the Vendor Command to the printer related to printer driver name, and acquire the response data from printer.

Parameters

lpPrinterDriverName

Specify the printer name allocated when the printer driver is installed.

lpVendorCmd

Pointer to the VENDOR_COMMAND structure.

lpRcvData

Pointer to the RECEIVE_DATA structure.

Return Value

= 1 : Success

≠ 1 : Failure

Remarks

The execution method are the same as the FcITP_GetVendorCommandEx function. Please refer to page 5 of this document

– Use of exported DLL functions –

- 1) Use LoadLibrary function to load the “FTPCtrl.DLL” or “FTPCtrl_x64.DLL”.
- 2) After the “FTPCtrl.DLL” or “FTPCtrl_x64.DLL” is loaded, call the GetProcAddress function to acquire the address of the exported DLL functions.
- 3) Call the exported DLL functions using the function pointers returned by GetProcAddress function.

– Example 1) Get the printer status by calling FcITP_GetVendorCommand function –

```
//Define the function prototype
typedef ULONG (__cdecl *pFcITP_Search_USB)(void);
typedef ULONG (__cdecl *pFcITP_GetVendorCommand)(PVENDOR_COMMAND pVendorCmd, PRECEIVE_DATA pRcvData);

HMODULE hDll;
VENDOR_COMMAND VendorCmd;
RECEIVE_DATA RcvData;

pFcITP_Search_USB fnFcITP_Search_USB;
pFcITP_GetVendorCommand fnFcITP_GetVendorCommand;

//Load the DLL and keep the handle to it
hDll = LoadLibrary("FTPCtrl.DLL");

// If the handle is valid, try to get the function address
if (hDll != NULL) {

    //Get pointer to our function using GetProcAddress
    fnFcITP_Search_USB = (pFcITP_Search_USB)GetProcAddress(hDll,"FcITP_Search_USB");
    fnFcITP_GetVendorCommand = (pFcITP_GetVendorCommand)GetProcAddress(hDll,"FcITP_GetVendorCommand");

    if ( (fnFcITP_Search_USB)() == 1 ) { // Search USB port driver

        VendorCmd.bRequest = 1;
        VendorCmd.wValueH = 0;        VendorCmd.wValueL = 0;
        VendorCmd.wIndexH = 0;        VendorCmd.wIndexL = 0;
        VendorCmd.wLengthH = 0;        VendorCmd.wLengthL = 4; //The length of data depends on the printer.
        VendorCmd.unitLength = sizeof(VENDOR_COMMAND);
        RcvData.unitLength = sizeof(RECEIVE_DATA);

        if ( (fnFcITP_GetVendorCommand)( &VendorCmd, &RcvData ) == 1 ) { // vendor request

            // The status data is preserved in RcvData.Data[0]~RcvData.Data[3].
            // The status of each bit depends on the printer.
            if ( RcvData.Data[2] & 0x04 ) {
                // Out of paper
                ...
            } else if ( RcvData.Data[1] & 0x04 ) {
                // Platen open
                ...
            }
        }
    }
}

//Free the library
FreeLibrary(hDll);
}
```


– Example 2) Get the printer status by calling FcITP_GetVendorCommandEx function –

It is a case when the printer names were allocated to be “FTP639U (576pixels)_1” and “FTP639U (576pixels)_2” when the printer drivers were installed.

```
//Define the function prototype
typedef ULONG (_cdecl *pFcITP_Search_USBEx)(LPSTR lpPrinterDriverName);
typedef ULONG (_cdecl *pFcITP_GetVendorCommandEx)(LPSTR lpPrinterDriverName,
                                                    PVENDOR_COMMAND pVendorCmd, PRECEIVE_DATA pRcvData);

HMODULE hDll;
VENDOR_COMMAND VendorCmd;
RECEIVE_DATA RcvData;
pFcITP_Search_USBEx fnFcITP_Search_USBEx;
pFcITP_GetVendorCommandEx fnFcITP_GetVendorCommandEx;

//Load the DLL and keep the handle to it
hDll = LoadLibrary("FTPctrl.dll");

// If the handle is valid, try to get the function address
if (hDll != NULL) {

    //Get pointer to our function using GetProcAddress
    fnFcITP_Search_USBEx = (pFcITP_Search_USBEx)GetProcAddress(hDll, "FcITP_Search_USBEx");
    fnFcITP_GetVendorCommandEx = (pFcITP_GetVendorCommandEx)GetProcAddress(hDll, "FcITP_GetVendorCommandEx");

    if ( (fnFcITP_Search_USBEx)("FTP639U (576pixels)_1") == 1 ) { // Search USB port driver
        VendorCmd.bRequest = 1;
        VendorCmd.wValueH = 0;          VendorCmd.wValueL = 0;
        VendorCmd.wIndexH = 0;          VendorCmd.wIndexL = 0;
        VendorCmd.wLengthH = 0;          VendorCmd.wLengthL = 4; //The length of data depends on the printer.
        VendorCmd.unitLength = sizeof(VENDOR_COMMAND);
        RcvData.unitLength = sizeof(RECEIVE_DATA);

        if ( (fnFcITP_GetVendorCommandEx)("FTP639U (576pixels)_1", &VendorCmd, &RcvData) == 1 ) { // vendor request send

            // The status data is preserved in RcvData.Data[0]~RcvData.Data[3].
            // The status of each bit depends on the printer.
            if ( RcvData.Data[2] & 0x04 ) {
                // Out of paper
                ...
            } else if ( RcvData.Data[1] & 0x04 ) {
                // Platen open
                ...
            }
        }
    }

    if ( (fnFcITP_Search_USBEx)("FTP639U (576pixels)_2") == 1 ) { // next printer

        VendorCmd.bRequest = 1;
        VendorCmd.wValueH = 0;          VendorCmd.wValueL = 0;
        VendorCmd.wIndexH = 0;          VendorCmd.wIndexL = 0;
        VendorCmd.wLengthH = 0;          VendorCmd.wLengthL = 4; //The length of data depends on the printer.
        VendorCmd.unitLength = sizeof(VENDOR_COMMAND);
        RcvData.unitLength = sizeof(RECEIVE_DATA);

        if ( (fnFcITP_GetVendorCommandEx)("FTP639U (576pixels)_2", &VendorCmd, &RcvData) == 1 ) { // vendor request send

            // The status data is preserved in RcvData.Data[0]~RcvData.Data[3].
            // The status of each bit depends on the printer.
            if ( RcvData.Data[2] & 0x04 ) {
                // Out of paper
                ...
            } else if ( RcvData.Data[1] & 0x04 ) {
                // Platen open
                ...
            }
        }
    }

    FreeLibrary(hDll); //Free the library
}
}
```

– **Example 3) Get Information on whether the print ended by calling FcITP_GetVendorCommand function –**
 /*It is case when it transmits reply parameter setting command (FS r n(=2)) after image data is transmitted.*/

```
//Define the function prototype
typedef ULONG (_cdecl *pFcITP_Search_USB)(void);
typedef ULONG (_cdecl *pFcITP_GetVendorCommand)(PVENDOR_COMMAND pVendorCmd, PRECEIVE_DATA pRcvData);

HMODULE hDll;
VENDOR_COMMAND VendorCmd;
RECEIVE_DATA RcvData;

pFcITP_Search_USB fnFcITP_Search_USB;
pFcITP_GetVendorCommand fnFcITP_GetVendorCommand;

//Load the DLL and keep the handle to it
hDll = LoadLibrary("FTPCtrl.DLL");

// If the handle is valid, try to get the function address
if (hDll != NULL) {

    //Get pointer to our function using GetProcAddress
    fnFcITP_Search_USB = (pFcITP_Search_USB)GetProcAddress(hDll, "FcITP_Search_USB");
    fnFcITP_GetVendorCommand = (pFcITP_GetVendorCommand)GetProcAddress(hDll, "FcITP_GetVendorCommand");

    if ( (fnFcITP_Search_USB)() == 1 ) { // Search USB port driver

        VendorCmd.bRequest = 1;
        VendorCmd.wValueH = 0;          VendorCmd.wValueL = 0;
        VendorCmd.wIndexH = 0;          VendorCmd.wIndexL = 0;
        VendorCmd.wLengthH = 0;          VendorCmd.wLengthL = 4; //The length of data depends on the printer.
        VendorCmd.unitLength = sizeof(VENDOR_COMMAND);
        RcvData.unitLength = sizeof(RECEIVE_DATA);

        while(1){
            // Repeats until the vendor request changes.
            if ( (fnFcITP_GetVendorCommand)(&VendorCmd, &RcvData) == 1 ) { // vendor request

                // The status data is preserved in RcvData.Data[0]~RcvData.Data[3].
                // The status of each bit depends on the printer.
                if ( RcvData.Data[3] == 0x02 ) {
                    // print end
                    ...
                    break;
                }
            }
        }
    }
}

//Free the library
FreeLibrary(hDll);
}
```

– Example 4) Get the command response by calling FcITP_GetVendorCommand function –

/*It is case when it transmits memory switch transmission (GS (E pL pH fn a (fn=4)).*/

```
//Define the function prototype
typedef ULONG (_cdecl *pFcITP_Search_USB)(void);
typedef ULONG (_cdecl *pFcITP_GetVendorCommand)(PVENDOR_COMMAND pVendorCmd, PRECEIVE_DATA pRcvData);

HMODULE hDll;
VENDOR_COMMAND VendorCmd;
RECEIVE_DATA RcvData;
int i = 0;
char text[256] = "";
char format[256] = "";

pFcITP_Search_USB fnFcITP_Search_USB;
pFcITP_GetVendorCommand fnFcITP_GetVendorCommand;

//Load the DLL and keep the handle to it
hDll = LoadLibrary("FTPCtrl.DLL");

// If the handle is valid, try to get the function address
if (hDll != NULL) {

    //Get pointer to our function using GetProcAddress
    fnFcITP_Search_USB = (pFcITP_Search_USB)GetProcAddress(hDll,"FcITP_Search_USB");
    fnFcITP_GetVendorCommand = (pFcITP_GetVendorCommand)GetProcAddress(hDll,"FcITP_GetVendorCommand");

    if ( (fnFcITP_Search_USB)() == 1 ) { // Search USB port driver

        VendorCmd.bRequest = 9;
        VendorCmd.wValueH = 0;          VendorCmd.wValueL = 0;
        VendorCmd.wIndexH = 0;          VendorCmd.wIndexL = 0;
        VendorCmd.wLengthH = 0;          VendorCmd.wLengthL = 10; //The length of data depends on the command.
        VendorCmd.unitLength = sizeof(VENDOR_COMMAND);
        RcvData.unitLength = sizeof(RECEIVE_DATA);

        if ( (fnFcITP_GetVendorCommand)( &VendorCmd, &RcvData ) == 1 ) { // vendor request

            for (i = 0; i < 8; i++) {
                strcat_s(format, 256, "0x%02X ");
            }

            sprintf_s(text, 256, format, RcvData.Data[1], RcvData.Data[2], RcvData.Data[3], RcvData.Data[4],
                RcvData.Data[5], RcvData.Data[6], RcvData.Data[7], RcvData.Data[8]);
            MessageBox(NULL, text, "Command response", MB_OK);
        }
    }
}

//Free the library
FreeLibrary(hDll);
}
```