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

Date 2008/2/15
Fujitsu Component Limited

The enclosed 'FTPCtrl.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.

– Exported DLL functions –

Exported DLL function	Effect
FclTP_Search_USB	Search USB port driver, and acquire the port driver handle.
FclTP_GetVendorCommand	Send Vender Commands to printer, and get response data from printer.
FclTP_Search_USBEx	Search USB port driver related to printer driver name and acquire the port driver handle.
FclTP_GetVendorCommandEx	Send Vender Commands to printer related to printer driver name and get response data from printer.

– Specifications of exported DLL functions –

ULONG FclTP_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 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.

ULONG FclTP_GetVendorCommand (PVENDOR_COMMAND *lpVendorCmd*, PRECEIVE_DATA *lpRcvData*)

Send Vender 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

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

```
// VENDOR_COMMAND structure
typedef struct _VENDOR_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 _RECEIVE_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

if FclTP_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 FclTP_Search_USBEx (LPTSTR *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 FclTP_GetVendorCommandEx (LPTSTR *lpPrinterDriverName*,
PVENDOR_COMMAND *lpVendorCmd*,
PRECEIVE_DATA *lpRcvData*)

Send the Vender 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

FclTP_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 FclTP_Search_USBEx function, and to acquire the handle of USB port driver.
When only the FclTP_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 FclTP_GetVendorCommand function. Please refer to page 2 and 5 of this document.

– Use of exported DLL functions –

1) Use LoadLibrary function to load the 'FTPCtrl.DLL' in the Windows folder.
2) After the 'FTPCtrl.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) Get the printer status by calling FclTP_GetVendorCommand function –

```
//Define the function prototype
typedef ULONG (*pFclTP_Search_USB)(void);
typedef ULONG (*pFclTP_GetVendorCommand)(PVENDOR_COMMAND pVendorCmd, PRECEIVE_DATA pRcvData);

HMODULE    hDll;
VENDOR_COMMAND VendorCmd;
RECEIVE_DATA RcvData;

pFclTP_Search_USB    fnFclTP_Search_USB;
pFclTP_GetVendorCommand    fnFclTP_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
    fnFclTP_Search_USB = (pFclTP_Search_USB)GetProcAddress(hDll,"FclTP_Search_USB");
    fnFclTP_GetVendorCommand = (pFclTP_GetVendorCommand)GetProcAddress(hDll,"FclTP_GetVendorCommand");

    if ( (fnFclTP_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;          //Response data size = 4bytes
        VendorCmd.unitLength = sizeof(VENDOR_COMMAND);
        RcvData.unitLength = sizeof(RECEIVE_DATA);

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

            // The status data is preserved in RcvData.Data[0]～RcvData.Data[3].
            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 FclTP_GetVendorCommandEx function –

It is a case when the printer names were allocated to be “FTP627U_1" and “FTP627U_2" when the printer drivers were installed.

```
//Define the function prototype
typedef ULONG (*pFclTP_Search_USBEx)(LPTSTR lpPrinterDriverName);
typedef ULONG (*pFclTP_GetVendorCommandEx)(LPTSTR lpPrinterDriverName,
                                                    PVENDOR_COMMAND pVendorCmd, PRECEIVE_DATA pRcvData);

HMODULE   hDll;
VENDOR_COMMAND   VendorCmd;
RECEIVE_DATA   RcvData;

pFclTP_Search_USBEx   fnFclTP_Search_USBEx;
pFclTP_GetVendorCommandEx   fnFclTP_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
    fnFclTP_Search_USBEx = (pFclTP_Search_USBEx)GetProcAddress(hDll, "FclTP_Search_USBEx");
    fnFclTP_GetVendorCommandEx = (pFclTP_GetVendorCommandEx)GetProcAddress(hDll, "FclTP_GetVendorCommandEx");

    if ( (fnFclTP_Search_USBEx)(“FTP627U_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;              //Response data size = 4bytes
        VendorCmd.unitLength = sizeof(VENDOR_COMMAND);
        RcvData.unitLength = sizeof(RECEIVE_DATA);

        if ( (fnFclTP_GetVendorCommandEx)(“FTP627U_1”, &VendorCmd, &RcvData) == 1 ) {     // vendor request send

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

    if ( (fnFclTP_Search_USBEx)(“FTP627U_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;              //Response data size = 4bytes
        VendorCmd.unitLength = sizeof(VENDOR_COMMAND);
        RcvData.unitLength = sizeof(RECEIVE_DATA);

        if ( (fnFclTP_GetVendorCommandEx)(“FTP627U_2”, &VendorCmd, &RcvData) == 1 ) {   // vendor request send

            // The status data is preserved in RcvData.Data[0]～RcvData.Data[3].
            if ( RcvData.Data[2] & 0x04 ) {
                // Out of paper
                    …
            } else if ( RcvData.Data[1] & 0x04 ) {
                // Platen open
                    …
            }
        }
    }
    //Free the library
    FreeLibrary(hDll);
}
```

– Example3) Get Information on whether the print ended by calling FclTP_GetVendorCommand function –

/*It is case when it transmits FS r command (0x1C 0x72 0x02) after image data is transmitted.*/

```
//Define the function prototype
typedef ULONG (*pFclTP_Search_USB)(void);
typedef ULONG (*pFclTP_GetVendorCommand)(PVENDOR_COMMAND pVendorCmd, PRECEIVE_DATA pRcvData);

HMODULE    hDll;
VENDOR_COMMAND VendorCmd;
RECEIVE_DATA RcvData;

pFclTP_Search_USB    fnFclTP_Search_USB;
pFclTP_GetVendorCommand    fnFclTP_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
    fnFclTP_Search_USB = (pFclTP_Search_USB)GetProcAddress(hDll,"FclTP_Search_USB");
    fnFclTP_GetVendorCommand = (pFclTP_GetVendorCommand)GetProcAddress(hDll,"FclTP_GetVendorCommand");

    if ( (fnFclTP_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;          //Response data size = 4bytes
        VendorCmd.unitLength = sizeof(VENDOR_COMMAND);
        RcvData.unitLength = sizeof(RECEIVE_DATA);

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

                 // The status data is preserved in RcvData.Data[0]～RcvData.Data[3].
                 if ( RcvData.Data[3] == 0x02 ) {
                     // print end
                     …
                     break;
                 }
             }
        }
    }
    //Free the library
    FreeLibrary(hDll);
}
```