AN1182
APPLICATION NOTE

USING THE ST7 USB LOW-SPEED FIRMWARE

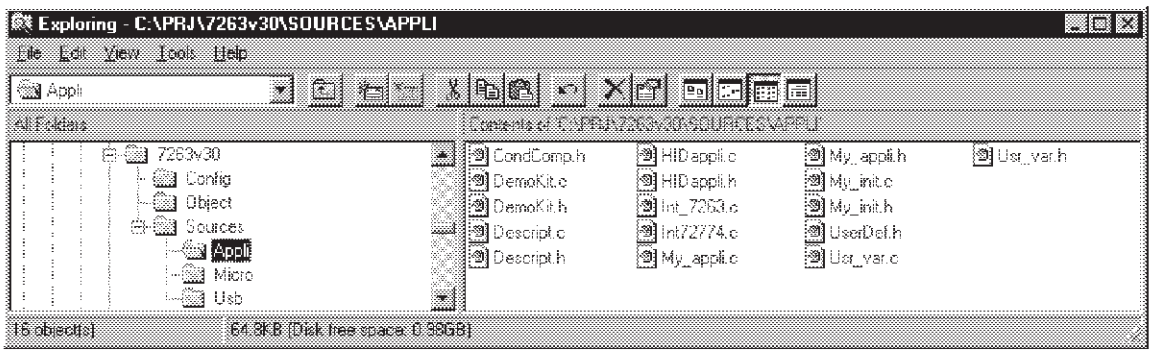
by Microcontroller Division Applications

This application note describes how to use the ST7_USB firmware. This firmware, written in C, using the Hiware C compiler, provides a complete USB protocol layer for low-speed USB microcontrollers (such as the ST7262, ST7263 and ST72774). The source code is available free to STMicroelectronics customers.

1 OVERVIEW

The firmware files, supplied in a ZIP file, are organised in the directories shown in Figure 1.

Figure 1. Directory Structure and Contents of SOURCES\APPLI



In order to use the ST7_USB_LOW_SPEED_FIRMWARE, you will only have to modify the files contained in the ROOT and APPLI directory. You will not need to touch any of the files in the other directories (MICRO and USB).

2 MODIFYING THE FILES IN THE ROOT DIRECTORY

You must choose the .mak and .prm files corresponding to the device you are using.
For example, if you are using a ST72774 with 60K of ROM, you have to open the file Demo72774.mak and make sure that all references to .prm files use the name 72774_60k.prm.

If you want to add some source files or directories, you have to customize them according to the Application Note AN989 titled “Starting With ST7 HIWARE C”.

3 MODIFYING THE FILES IN THE APPLI DIRECTORY

All the application-specific files have to be located in the APPLI directory.

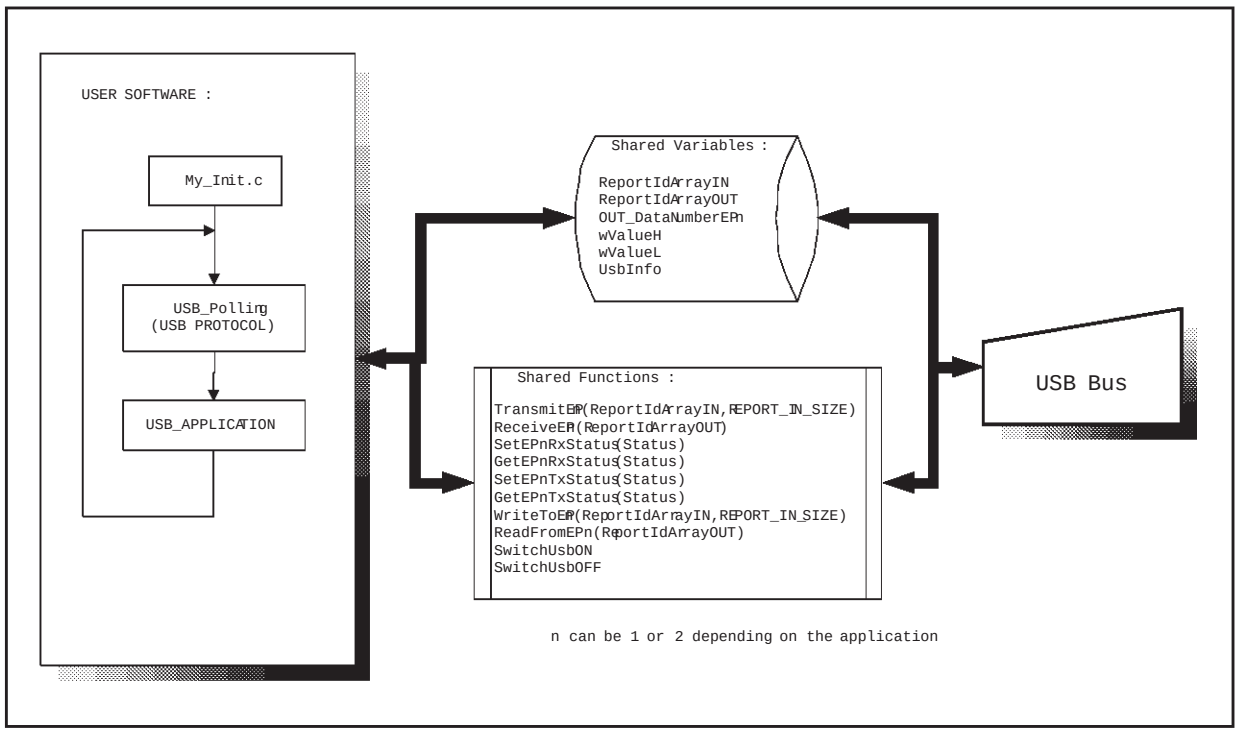
Here is a description of the files that are in the directory initially:

File Name	Description
Condcomp.h	Compiler Directives: product selection, clock frequency, LVD, Watchdog, etc....
Descript.c and .h	Descriptor files: you have to modify them according to your application (the initial values filled in apply to the ST7263DEMOKIT)
Int_7263.c and .h or Int72774.c and .h	All the interrupt routines for your application have to be in one of these files (depends on the selected microcontroller).
My_appli.c and .h	Your application has to be in this file with the name: ”USB_APPLICATION”. Some HID class functions are also available in this file. The My_suspend and My_resume functions have to be completed in this file.
My_init.c and .h	Your initialisation routine.
usr_var.c and .h	Declaration of all your application variables.
HID_appli.c and .h	HID specific functions (like GetReport &SetReport)

4 EXPLANATION OF VARIABLES AND FUNCTIONS

As shown in Figure 2, the USB protocol firmware, (which you do not have to modify) uses a certain number of variables and functions which you can also use in your application source files (Directory /Appli). These variables and functions are also used in the data transfer functions and interrupt routines (Int_USB.c).

Figure 2. IInterdependancy of USB Variables and Functions



4.1 VARIABLES

Here is a description of the variables.The first one, ReportIdArray, is declared in the usr_var.c file, the others are declared in the usb_var.c file:

Variable	Description
ReportIdArray_IN ReportIdArray_OUT	Arrays used in the application index 0 => Report Id Number index 1 to Report_IN_Size (or Report_OUT_size) => Data report
REPORT_IN_SIZE REPORT_OUT_SIZE	Number of data bytes to exchange with the host
OUT_DataNumberEP*n*	Number of data bytes received on Endpoint number *n*
wValueH	Class Report type used in SetReport & GetReport
wValueL	Report Id Number, used in GetReport function
UsbInfo	Select product in multiple product & mode

4.2 DATA TRANSFER FUNCTION CALLS

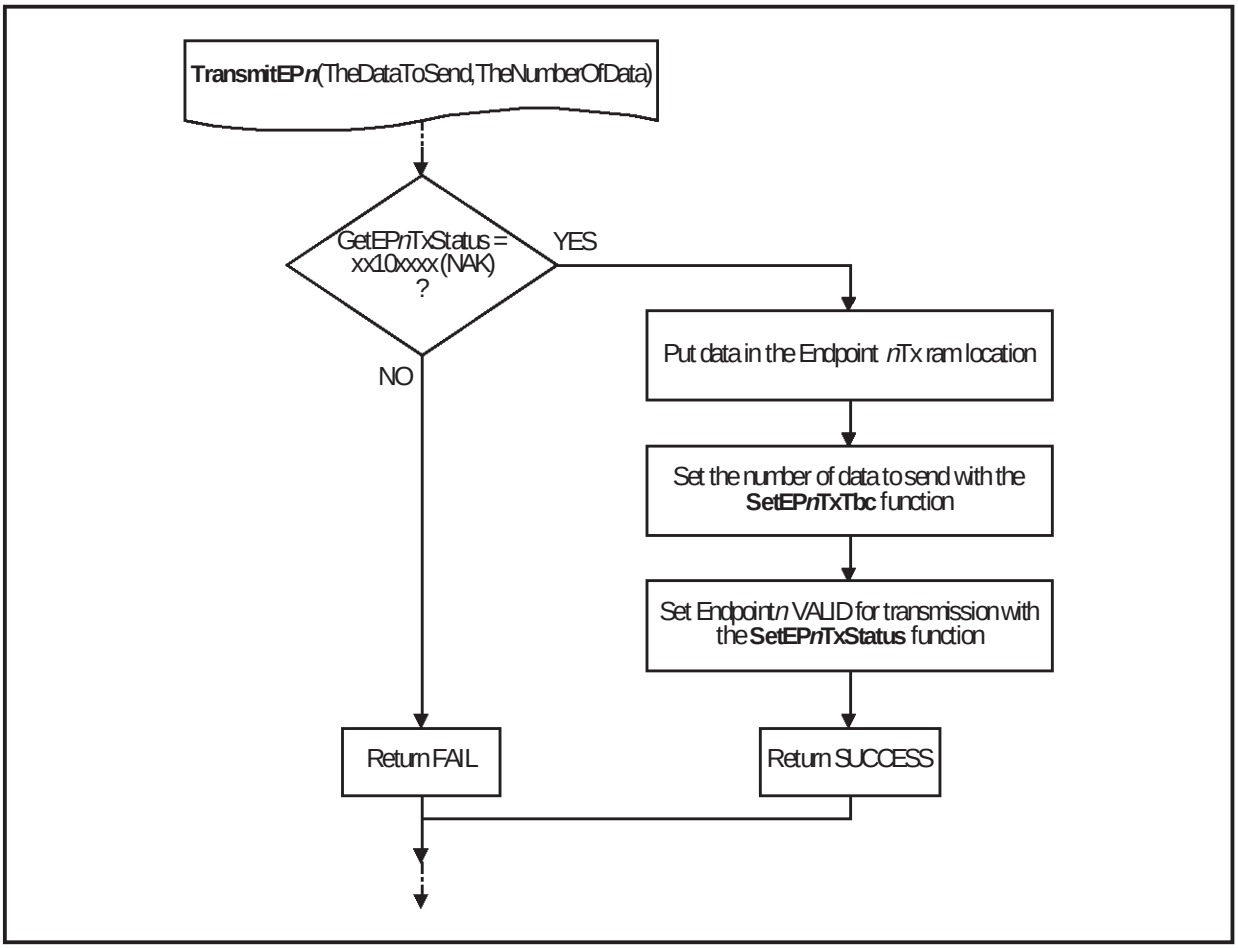
To transfer data over the USB, include any of the function calls below in your application program. Figure 3, Figure 4 and Figure 5 give flowcharts showing their use.

Function	Description
TransmitEP*n*(ReportIdArrayIN,REPORT_IN_SIZE)	Send data contained in ReportIdArrayIN from device to Host PC via Endpoint number *n*
ReceiveEP*n*(ReportIdArrayOUT)	Put in ReportIdArrayOUT, the data received from the Host PC via Endpoint *n*
SetEP*n*RxStatus(Status)	In reception, set Endpoint *n* to VALID, NAK, STALL or DISABLE
GetEP*n*RxStatus(Status)	In reception, get the status of Endpoint *n*
SetEP*n*TxStatus(Status)	In transmission, set Endpoint *n* to VALID, NAK, STALL or DISABLE
GetEP*n*TxStatus(Status)	In transmission, get the status of Endpoint *n*
WriteToEP*n*(ReportIdArrayIN,REPORT_IN_SIZE)	Put data on the USB Buffer, ready to be sent, wait the validation of the EP*n* in transmission
ReadFromEP*n*(ReportIdArrayOUT)	Read data from the EP*n* buffer, the EP*n* still unvalidated after the read.
SwitchUsbON	Switch ON the ST7 on-chip USB interface
SwitchUsbOFF	Switch OFF the ST7 on-chip USB interface

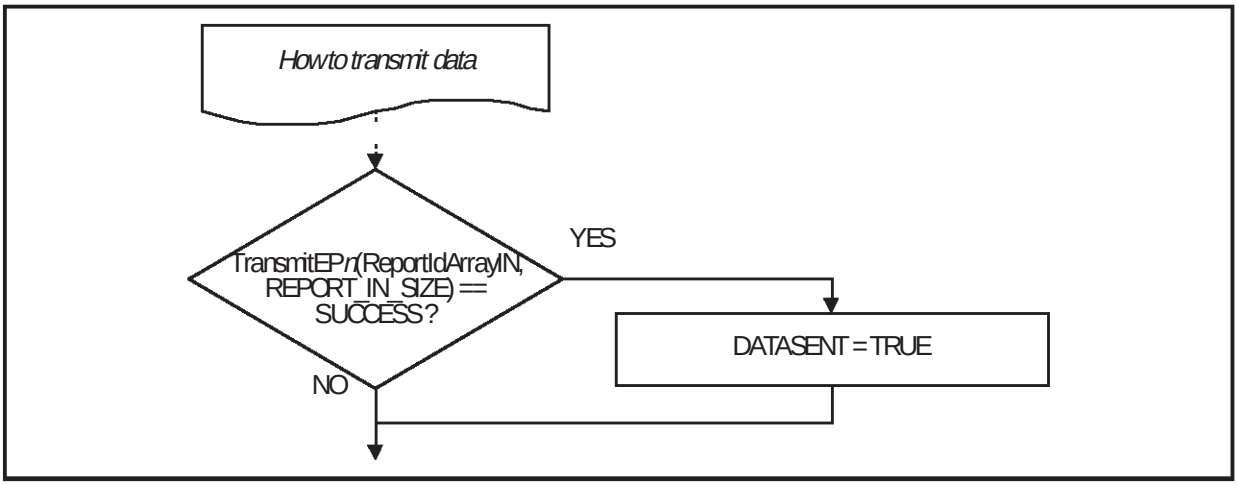
4.2.1 Sending Data

To send data, the TransmitEP*n* function must be used.

Figure 3. TransmitEP*n*(TheDataToSend,TheNumberOfData)



Here is an example of the way the TransmitEP*n*() function has to be used:



4.2.2 . Receiving Data

Figure 4. ReceiveEP*n*(TheDataReceived)

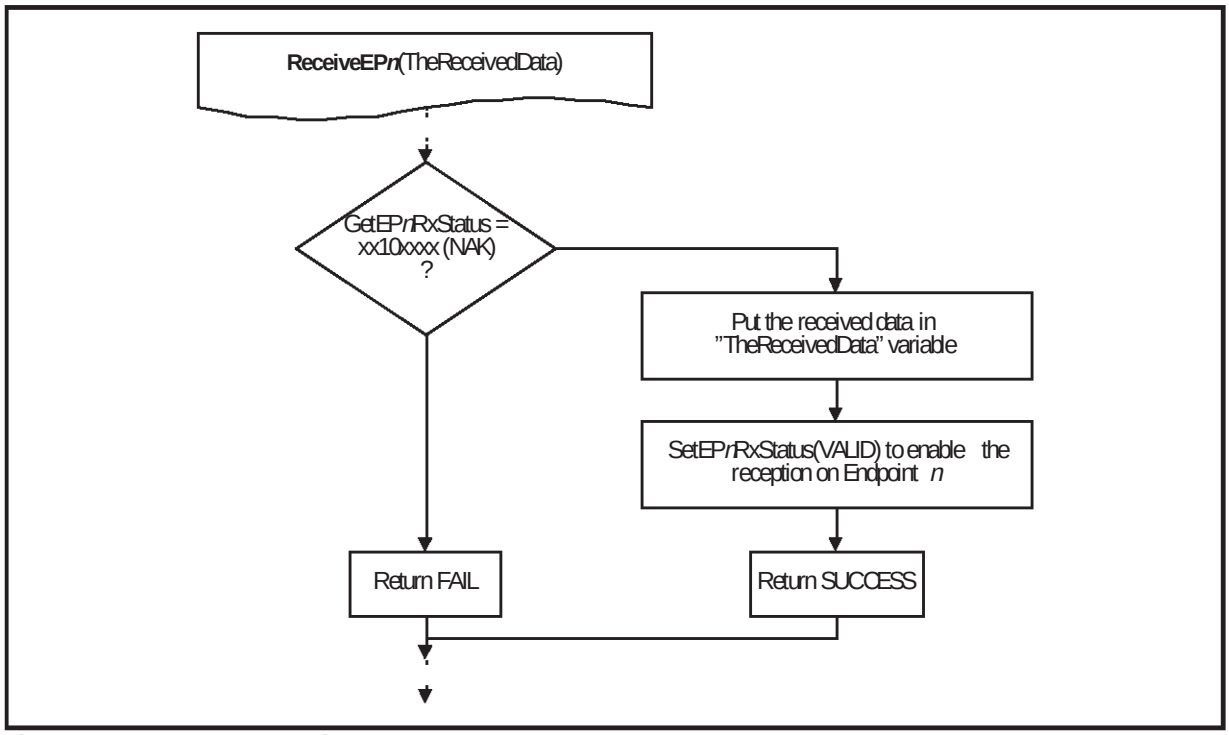
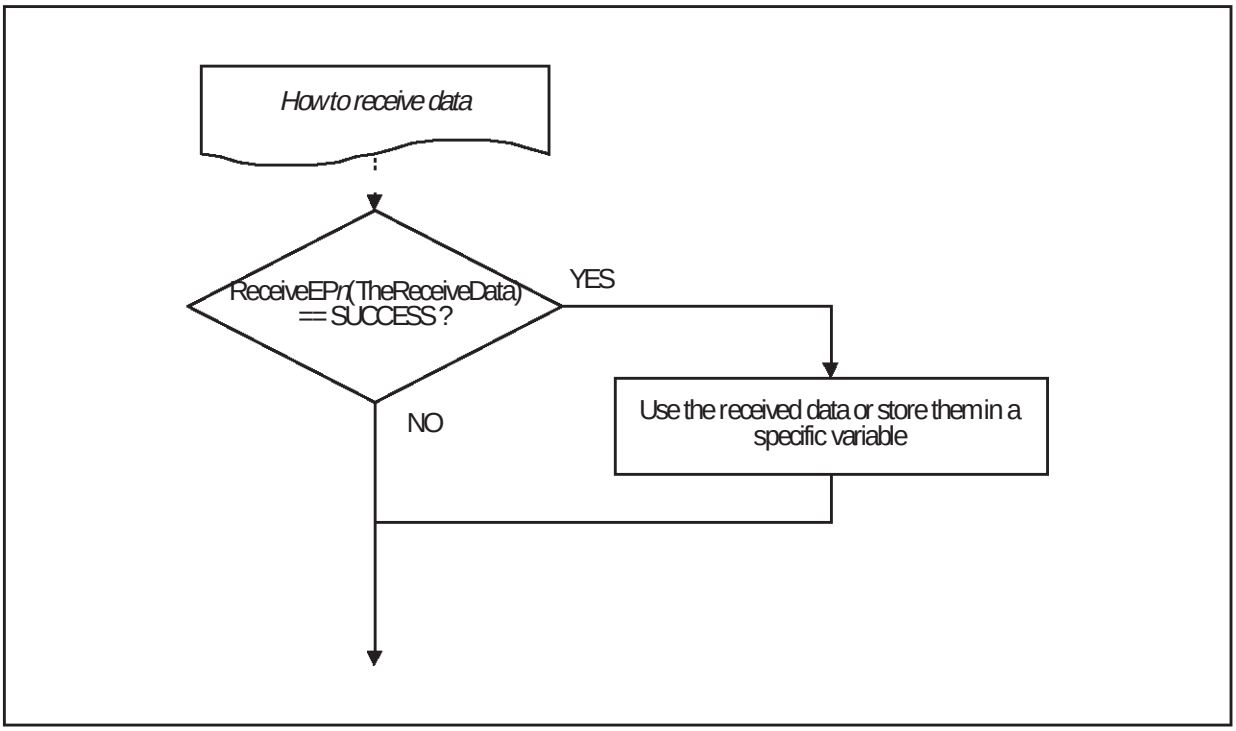


Figure 5. How To Receive Data



4.3 FUNCTIONS ALREADY AVAILABLE FOR HID CLASS (IN MY_APPLI.C)

You will find the following functions which have been defined in My_appli.c.

Function	Description
BUILD_FEATURE_REPORT	Get a specific feature of the device
BUILD_OUTPUT_REPORT	Get a specific value for an Output feature
GetReport	Select the right function to call between GetFeature and GetOutput.
PROCESS_FEATURE_REPORT	Set a specific feature of the device
PROCESS_OUTPUT_REPORT	Set a specific Output feature of the device
SetReport	Select the Set Report Type

Examples:

For the ST7263 demo kit, the value of the button is linked to ReportID 2, so the result of a GetReport(2) is the button value.

In the ST7263 demo kit, the ReportID of the PWM is 4, so a SetReport[4,40] will set the ST7263 PWM to a 40% duty cycle.