



## AN2405 Application note

### Migrating from RS232-only applications to RS232/USB applications using Turbo Plus uPSD

#### Introduction

Most laptops and recent desktops do not have serial ports however many legacy applications still use RS232. The scope of this application note is to migrate such RS232 applications to RS232/USB applications using Turbo Plus uPSD. Since Turbo Plus uPSD has two UART ports, one of them is used for the USB-to-Serial Bridge connection. The scope is to implement an easy migration with minimal impact on legacy applications.

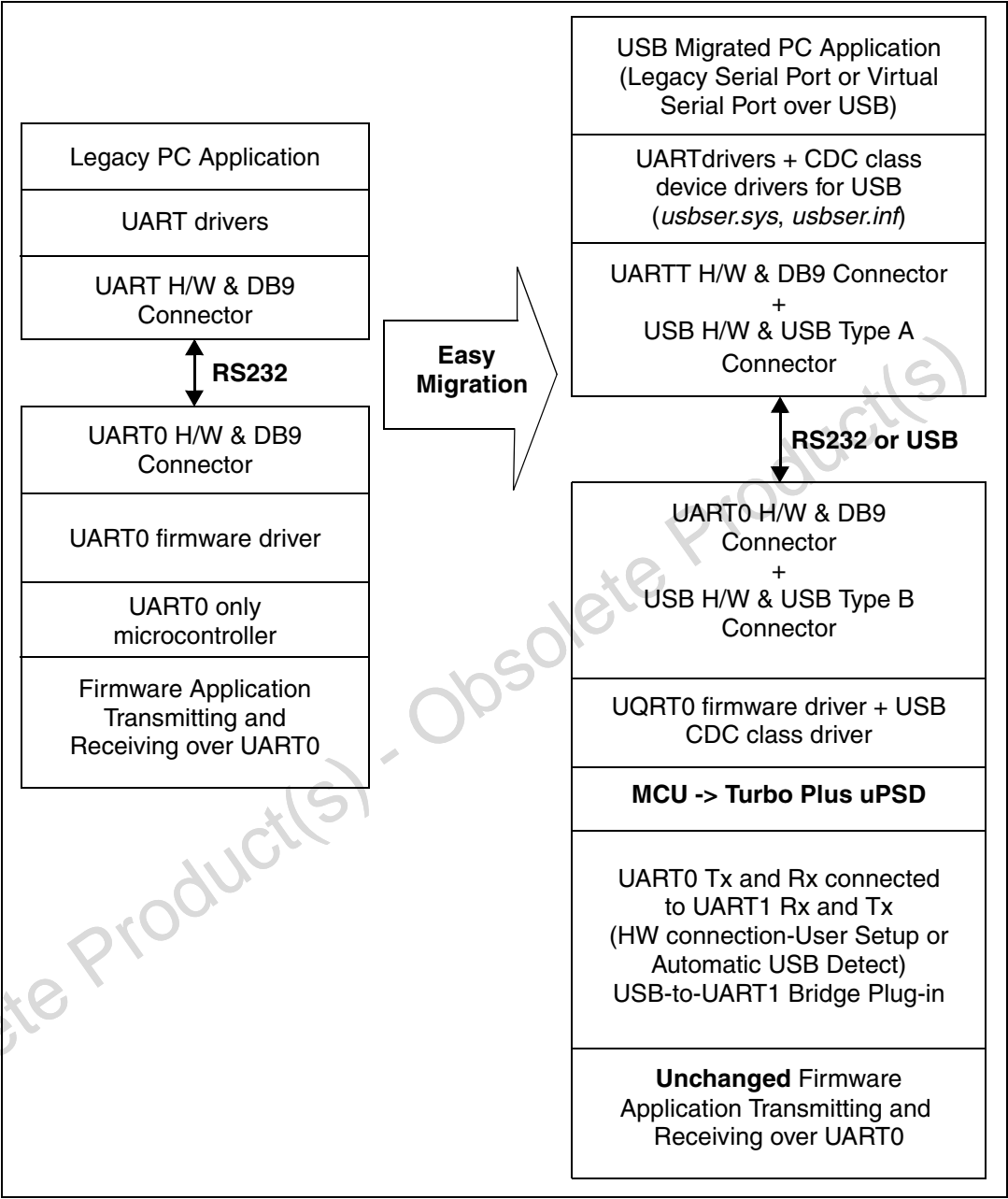
Obsolete Product(s) - Obsolete Product(s)

# Contents

<b>1</b>	<b>Migration description</b>	<b>3</b>
1.1	Architecture	3
1.2	Loading CDC class drivers for USB	3
1.3	.INF file	4
1.3.1	Usbser.inf (Windows 2000,XP only) file:	4
1.4	CDC Firmware Driver	5
1.5	CDC Class Requests	7
1.6	USB_UART1 Bridge (HW Connection-User setup or Automatic USB Detect)	8
1.7	USB-to-Serial Bridge plug-in	9
1.8	Example of easy migration to RS232_USB application:	9
<b>2</b>	<b>Conclusion</b>	<b>10</b>
<b>3</b>	<b>References</b>	<b>11</b>
<b>4</b>	<b>Revision history</b>	<b>12</b>

# 1 Migration description

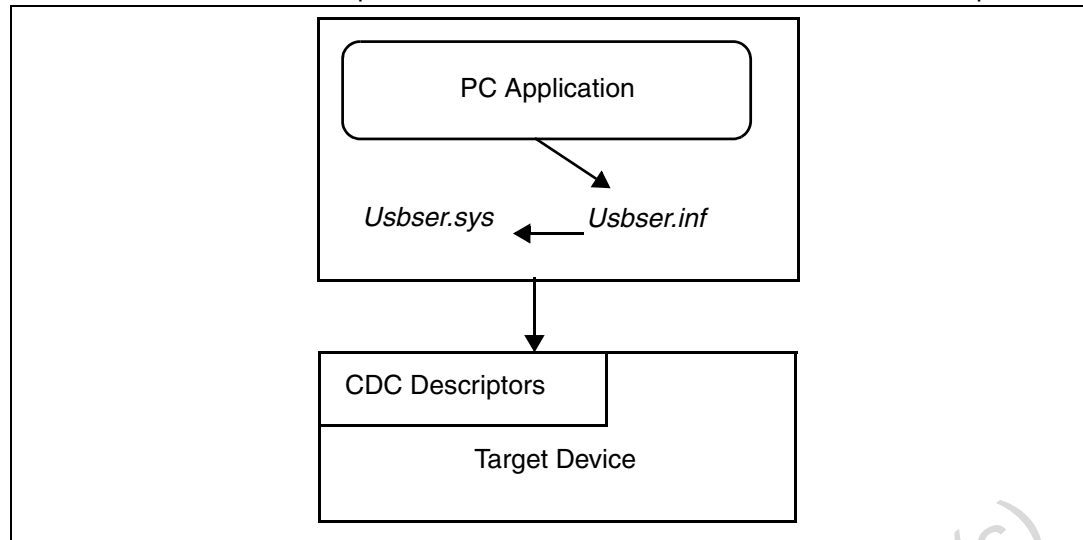
## 1.1 Architecture



## 1.2 Loading CDC class drivers for USB

Install the setup application provided with this example and the necessary CDC class drivers will be loaded on your PC. The driver installed will work with Windows 2000 & XP

only. The *.inf* file has to be modified to work with Windows 98/Me. When connected to a device that has CDC class implemented, Windows will see USB as a virtual COM port.



### 1.3 .INF file

When a USB device is connected, windows checks the *.inf* file to load the appropriate driver for the device.

Microsoft does not provide an *.inf* file with the *usbser.sys* driver file. The *.inf* file *usbser.inf* provided via the setup application allows the user to load the *usbser.sys* driver file under Windows 2000 and Windows XP. The *.inf* file has to be modified to work with Windows 98/Me.

Application manufacturers can modify this file and the device descriptors in *usb\_desc.h* for their own VID & PID.

To install the drivers from a directory instead of running the provided setup application, copy the *usbser.inf* file and *usbser.sys* file to a folder. When the dialog box displaying “Found New Hardware Wizard” opens up, choose **Install from a specific location** and point to the folder where the *usbser.inf* file is copied.

**Note:** Do not copy the *.inf* file into the *windows\inf* folder manually.

#### 1.3.1 Usbser.inf (Windows 2000,XP only) file:

```

[Version]
DriverVer=08/03/2004,5.1.2600.2180
Signature=$CHICAGO$
Class=Ports
ClassGUID={4d36e978-e325-11ce-bfc1-08002be10318}
Provider=%String0%

; Port Class Install
[ClassInstall]
AddReg=USBCClassInstall

[USBCClassInstall]
HKR,,, %Enumerator%
HKR, , Icon, , "-23 "
  
```

```

HKR,,Installer32,, "MsPorts.Dll, PortsClassInstaller"

[Manufacturer]
%String0%=STMicroelectronics

[ControlFlags]
ExcludeFromSelect=USB\VID_0483&PID_3420
ExcludeFromSelect=USB\VID_0483&PID_3400

[STMicroelectronics]
%String1%=STUSB, USB\VID_0483&PID_3401

; Win 2000
[STUSB]
;CopyFiles=WDM.Drvrs
AddReg=STUSB.AddReg, DevMap.AddReg

; [WDM.Drvrs]
; usbser.sys,, 0x00000020

[STUSB.AddReg]
HKR,,DevLoader,0,*ntkern
HKR,,NTMPDriver,, "usbser.sys"
HKR,,EnumPropPages32,, "MsPorts.dll, SerialPortPropPageProvider"

[DevMap.AddReg]
; HKLM, hardware\devicemap\serialcomm, COM5,, COM5

[STUSB.Services]
AddService=usbser, 0x00000002, FuncDrv_Service_Inst

[FuncDrv_Service_Inst]
DisplayName=%USBFilterString%
ServiceType= 1
StartType = 3
ErrorControl = 0
ServiceBinary = %12%\usbser.sys

[Strings]
String0="STMicroelectronics"
String1="STMicroelectronics USB Serial Emulator"
Enumerator = "USB Serial Emulator"
%USBFilterString% = "USB Serial Service"

```

## 1.4 CDC Firmware Driver

The device enumerates as a CDC driver and talks to the PC over USB on Virtual COM port. To be considered as a COM port, the USB device declares two interfaces:

- Abstract Control Model Communication class interface using interrupt endpoint 2(IN).
- Data Class Interface using Bulk endpoint 1(IN & OUT)

Descriptors (Device, Configuration and Interface):

```

const device_descriptor code deviceDesc =
{
    sizeof(device_descriptor),           // Size of this Descriptor in Bytes
    DT_DEVICE,                          // Descriptor Type (=1)
    {USB_SPEC_REV_LSB, USB_SPEC_REV_MSB}, // USB Spec Release Number in BCD
    CDC_CLASS,                          // Device Class Code (CDC class code)
    0,                                  // Device Subclass Code (CDC Subclass Code)
    0,                                  // Device Protocol Code (CDC Device Protocol)

```

```

    EP0_PKT_SIZE,                // Maximum Packet Size for EP0
{
    VENDOR_ID_LSB, VENDOR_ID_MSB }, // Vendor ID
{PRODUCT_ID_LSB, PRODUCT_ID_MSB}, // Product ID
{
    0x01, 0x02},
0,                                // Index of String Desc for Manufacturer
0,                                // Index of String Desc for Product
0,                                // Index of String Desc for SerNo
1                                  // Number of possible Configurations
};

const uchar code configDesc[] =
{
    9,                            // Configuration descriptor length
    2,                            // Descriptor type (configuration)
    0x43,                         // Total length of this descriptor 2EP+Control
    0x00,
    NO_OF_INTERFACES,            // Number of interfaces
    0x01,                         // Configuration value (OnSetConfiguration)
    0x00,                         // Index of string descriptor (none)
    BM_ATTRIBUTES,
    0x32,                         // 100 mA max power consumption

//Communication class Interface Descriptor Requirement
    0x09,                         // Descriptor length
    0x04,                         // Descriptor type (interface)
    0x00,                         // Number of interface
    0x00,                         // Alternate setting
    0x01,                         // Number of endpoints (except EP0)
    CDC_CLASS,                   // Class code - CDC class code
    ACM_VIRTUALCOM,              // Subclass - Abstract Control Model(for virtual COM)
    INTERFACE_PROTOCOL,          // Protocol - 0(interface protocol)
    0x00,                         // iInterface String

//Header functional descriptor
    0x05,                         //bFunctionLength
    0x24,                         //bDescriptor type - CS_INTERFACE
    0x00,                         //bDescriptor Subtype - Header Func Desc
    0x10,                         //bcd CDC spec release number
    0x01,                         //Call Management functional descriptor
    0x05,                         //bFunctionLength
    CS_INTERFACE,                //bDescriptor type - CS_INTERFACE
    CALL_MANAGEMENT_FN,          //bDescriptor subtype - Call Management Func Desc
    0x00,                         //bmCapabilities D0+D1
    0x01,                         //bDataInterface: 1
//ACM Functional descriptor
    0x04,                         //bFunction Length
    CS_INTERFACE,                //bDescriptor type - CS_INTERFACE
    ACM_VIRTUALCOM,              //bDescriptor subtype: ACM func desc
    0x02,                         //bmCapabilities

//Union Functional descriptor
    0x05,                         //bFunctionLength
    CS_INTERFACE,                //bDescriptor type - CS_INTERFACE
    UNION_MANAGEMENT_FN,         //bDescriptor subtype: Union func desc
    0x00,                         //bMasterInterface: Communication class interface
    0x01,                         //bSlaveInterface0: Data Class Interface

//Endpoint 2 descriptor
/* EP2 */
    0x07,                         // Descriptor length (7 bytes)
    0x05,                         // Descriptor type (endpoint)

```

```

0x82, // Address (IN3)
EP_ATTR_INTERRUPT, // Attributes-Interrupt transfer
0x08, 0, // Maximum packet size
0xFF, // Polling interval-Maximum

//Data class interface descriptor requirement
0x09, //bLength,
0x04, //bDescriptorType
0x01, //bInterface Number
0x00, //bAlternate setting
0x02, //bNumEndpoints
DATA_CLASS_INTERFACE, //bInterfaceClass-Data class interface
0x00, //bInterfaceSubclass
0x00, //bInterfaceProtocol
0x00, //iInterface

//Endpoint 1 Descriptor
/* EP1-OUT */
0x07, // Descriptor length (7 bytes)
0x05, // Descriptor type (endpoint)
0x01, // Address (OUT1)
EP_ATTR_BULK, // Attributes -bulk transfer
EP1_PKT_SIZE, 0, // Maximum packet size
0, // Polling interval not used

/* EP1-IN*/
0x07, // Descriptor length (7 bytes)
0x05, // Descriptor type (endpoint)
0x81, // Address (OUT1)
EP_ATTR_BULK, // Attributes -bulk transfer
EP1_PKT_SIZE, 0, // Maximum packet size
0 // Polling interval not used
};

```

## 1.5 CDC Class Requests

The firmware is required to handle the following CDC class requests:

Request	Code	Description	Req'd/Opt
SEND_ENCAPSULATED_COMMAND	00h	Issues a command in the format of the supported control protocol	Required
GET_ENCAPSULATED_RESPONSE	01h	Requests a response in the format of the supported control protocol	Required
SET_COMM_FEATURE	02h	Controls the settings for a particular communication feature	Optional
GET_COMM_FEATURE	03h	Returns the current settings for the communication feature	Optional
CLEAR_COMM_FEATURE	04h	Clears the settings for a particular communication feature	Optional
SET_LINE_CODING	20h	Configures DTE rate, stop-bits, parity, and number-of-character bits	Optional <sup>1)</sup>
GET_LINE_CODING	21h	Requests current DTE rate, stop-bits, parity, and number-of-character bits	Optional <sup>1)</sup>

Request	Code	Description	Req'd/Opt
SET_CONTROL_LINE_STATE	22h	RS-232 signal used to tell the DCE device the DTE device is now present	Optional
SEND_BREAK	23h	Sends special carrier modulation used to specify RS-232 style break	Optional

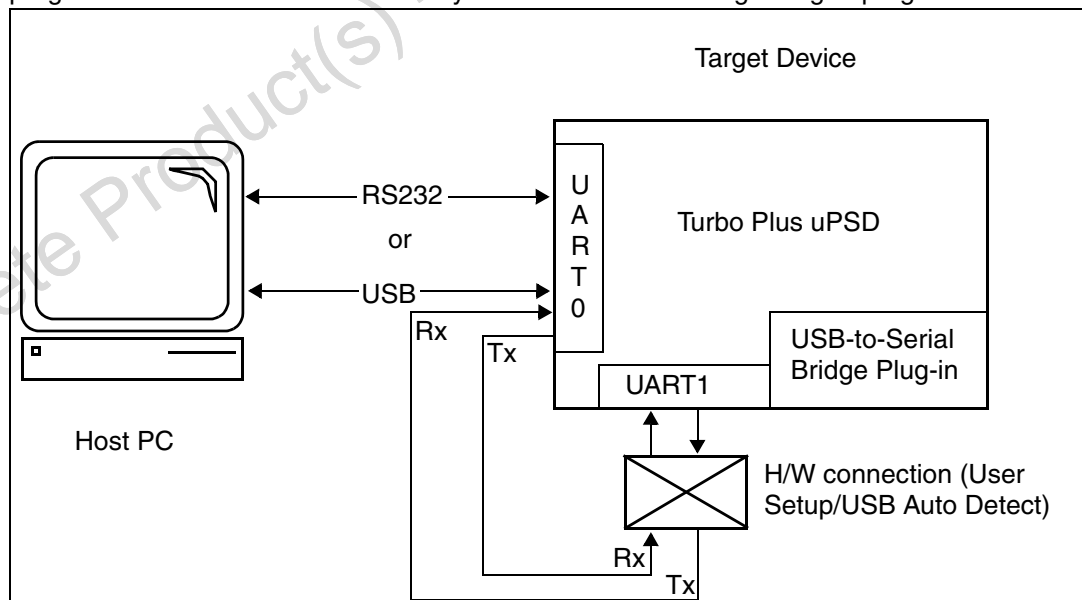
**Note:** 1 These requests can be used to configure the UARTs in USB-to Serial bridge implementations.

## 1.6 USB\_UART1 Bridge (HW Connection-User setup or Automatic USB Detect)

The target MCU for the migrated application is the Turbo Plus uPSD. The Turbo Plus uPSD has two UARTs - UART0 and UART1 and a USB communication interface. The UARTs can be configured using Timer2 to operate at one of the following Kbps baud rates:

f <sub>osc</sub> MHz	Desired Baud Rate	Timer 2 SFRs		Resulting Baud Rate	Baud Rate Deviation
		RCAP2H (hex)	RCAP2L(hex)		
40.0	115200	FF	F5	113636	-1.36%
40.0	57600	FF	EA	56818	-1.36%
40.0	28800	FF	D5	29070	0.94%
40.0	19200	FF	BF	19231	0.16%
40.0	9600	FF	7E	9615	0.16%

The Tx and Rx lines of UART0 must be connected to the Tx and Rx lines of UART1 when using Virtual COM port. This will enable the data transferred over UART0 by the application program to be transferred over USB by the USB-to Serial Bridge Plug-in program.



This connection can be done by the user or can be done through automatic USB detection.



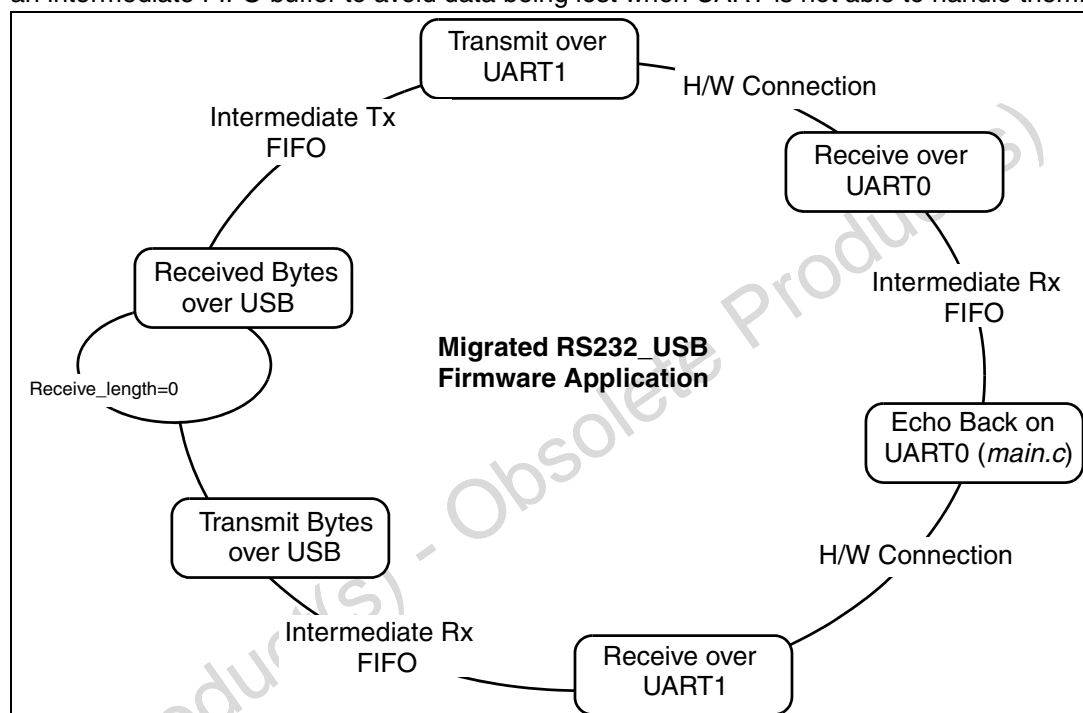
## 1.7 USB-to-Serial Bridge plug-in

The USB-to-Serial Bridge Plug-in is provided as a library file (upsd3400\_USB-to\_Serial.lib). Features of this library are:

- Small code footprint of approximately 3.5K
- Xdata memory space of 800 bytes including intermediate FIFO buffers used for transfer
- Data space of 50 bytes

## 1.8 Example of easy migration to RS232\_USB application:

The scope is to provide an easy migration. The example application echoes back character typed and sent to Turbo Plus uPSD. All received and transmitted characters are stored in an intermediate FIFO buffer to avoid data being lost when UART is not able to handle them.



## 2 Conclusion

The example application is aimed at easy migration from RS232 only applications to RS232/USB applications. There is a overhead of one UART port and the firmware is not efficient in RS232 transfers. Once the reader has familiarized himself with the concepts of USB and CDC Class implementation it is recommended that he moves to implement a USB-to Serial bridge.

Obsolete Product(s) - Obsolete Product(s)

### 3 References

1. Universal Serial Bus definitions for Communication devices, Version 1.1
2. uPSD34xx-Turbo Plus Series Fast Turbo 8032 MCU with USB and Programmable Logic
3. DK3400\_USB\_To\_Serial\_Bridge\_Using\_UART1 firmware.

Obsolete Product(s) - Obsolete Product(s)

## 4 Revision history

**Table 1. Document revision history**

Date	Revision	Changes
02-Jul-2007	1	Initial release.

Obsolete Product(s) - Obsolete Product(s)

**Please Read Carefully:**

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

**UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.**

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2007 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

[www.st.com](http://www.st.com)