USB Device TEC Scanner Class Driver User Guide

Version 1.20

For use with USBD TEC Scanner Class Driver versions 1.01 and above

Exported on 03/08/2019

All rights reserved. This document and the associated software are the sole property of HCC Embedded. Reproduction or duplication by any means of any portion of this document without the prior written consent of HCC Embedded is expressly forbidden.

HCC Embedded reserves the right to make changes to this document and to the related software at any time and without notice. The information in this document has been carefully checked for its accuracy; however, HCC Embedded makes no warranty relating to the correctness of this document.

Table of Contents

1 System Overview

This chapter contains the fundamental information for this module.

The component sections are as follows:

- Introduction – describes the main elements of the module.
- Feature Check – summarizes the main features of the module as bullet points.
- Packages and Documents – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- Change History – lists the earlier versions of this manual, giving the software version that each manual describes.

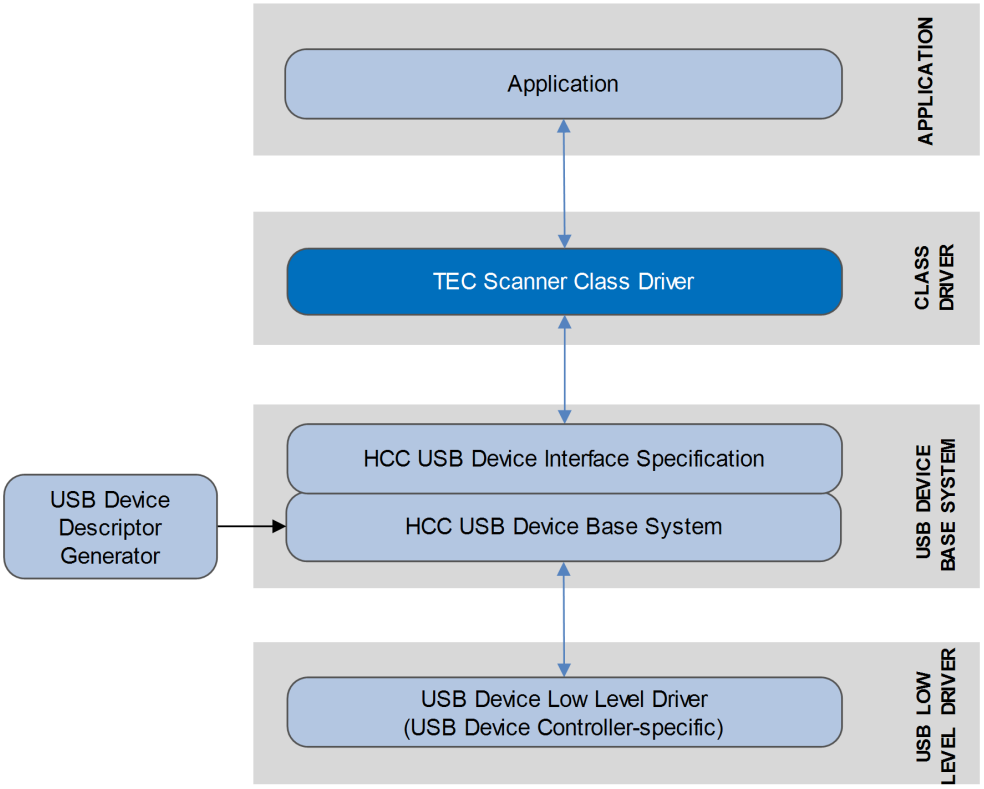
Note: To download this manual as a PDF, see USB Device PDFs.

1.1 Introduction

This guide is for those who want to emulate a Toshiba USB scanner device. TEC scanners, produced by Toshiba TEC Corporation, are used to scan bar codes.

The **usbd_cd_tec_scanner** package is a function device implementation of this proprietary USB class. This allows a device to connect to a host system and appear to it as a TEC scanner device.

The system structure is shown in the diagram below:



This class driver is effectively a library. It provides a set of function calls that an application can use to send and receive scanner data through the interface. The **usbd_tec_scanner_init()** function registers the class driver with the Embedded USB Device (EUSBD) base system and this call sets up callbacks for the base system to use.

Note: This module is part of HCC's EUSBD system, as described in the *HCC Embedded USB Device Base System User Guide*. This module communicates with the EUSBD base system through the EUSBD Device Interface, as described in the above manual.

1.2 Feature Check

The main features of the class driver are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Emulates a TEC Scanner device produced by Toshiba TEC Corporation.
- Compatible with sample device files produced by using HCC's *USB Device Descriptor Generator*.
- Allows the user to specify a callback for state change events.

1.3 Packages and Documents

Packages

This table lists the packages that you need in order to use this module:

Package	Description
hcc_base_doc	This contains the two guides that will help you get started.
usbd_base	The USB device base package. This is the framework used by USB class drivers to communicate over USB using a specific USB device controller package.
usbd_cd_tec_scanner	The USB device TEC scanner class driver package described by this document.

Documents

For an overview of HCC's embedded USB stacks, see Product Information on the main HCC website.

Readers should note the points in the HCC Documentation Guidelines on the HCC documentation website.

HCC Firmware Quick Start Guide

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

HCC Source Tree Guide

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

HCC Embedded USB Device Base System User Guide

This document defines the USB device base system upon which the complete USB stack is built.

HCC USB Device TEC Scanner Class Driver User Guide

This is this document.

HCC USB Device Descriptor Generator User Guide

This document describes the tool that creates USB descriptor files for inclusion in a project that uses the EUSBD stack.

1.4 Change History

This section describes past changes to this manual.

- To view or download manuals, see USB Device PDFs.
- For the history of changes made to the package code itself, see History: usbd_cd_tec_scanner.

The current version of this manual is 1.20. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.20	2019-03-08	1.01	New template.
1.10	2017-06-16	1.01	New *Change History* format.
1.00	2016-04-14	1.01	First release.

2 Source File List

This section describes all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the *HCC Source Tree Guide*. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

> **Note:** Do not modify any files except the configuration file.

2.1 API Header File

The file **src/api/api_usbd_tec_scanner.h** is the only file that should be included by an application using this module. For details of the API functions, see Application Programming Interface.

2.2 Configuration File

The file **src/config/config_usbd_tec_scanner.h** contains all the configurable parameters of the system. Configure these as required. For details of these options, see Configuration Options.

2.3 Source Code

The file **src/usb-device/class-drivers/tec_scanner/usbd_tec_scanner.c** contains the main code for the class driver. **This file should only be modified by HCC**.

2.4 Version File

The file **src/version/ver_usbd_tec_scanner.h** contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

3 Configuration Options

Set the system configuration options in the file **src/config/config_usbd_tec_scanner.h**. This section lists the available options and their default values.

USBD_TEC_SCANNER_MAX_BARCODE_LENGTH

The maximum length of the barcode to send. The default is 32.

USBD_TEC_SCANNER_TASK_STACK_SIZE

The task stack size. The default is 1024.

4 Application Programming Interface

This section documents the Application Programming Interface (API). It includes all the functions that are available to an application program.

4.1 Module Management

The functions are the following:

Function	Description
usbd_tec_scanner_init()	Initializes the module and allocates the required resources.
usbd_tec_scanner_start()	Starts the module.
usbd_tec_scanner_stop()	Stops the module.
usbd_tec_scanner_delete()	Deletes the module and releases the resources it used.

usbd_tec_scanner_init

Use this function to initialize the class driver and allocate the required resources.

> **Note:** You must call this before any other function.

Format

```
t_usbd_tec_scanner_ret usbd_tec_scanner_init ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBD_TEC_SCANNER_SUCCESS	Successful execution.
USBD_TEC_SCANNER_ERR_RESOURCE	Resource (mutex, event, or task) allocation error.

usbd_tec_scanner_start

Use this function to start the class driver.

> **Note:** You must call **usbd_tec_scanner_init()** before this to initialize the module.

Format

```
t_usbd_tec_scanner_ret usbd_tec_scanner_start ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBD_TEC_SCANNER_SUCCESS	Successful execution.
USBD_TEC_SCANNER_ERROR	Operation failed.

usbd_tec_scanner_stop

Use this function to stop the class driver.

Format

```
t_usbd_tec_scanner_ret usbd_tec_scanner_stop ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBD_TEC_SCANNER_SUCCESS	Successful execution.
USBD_TEC_SCANNER_ERROR	Operation failed.

usbd_tec_scanner_delete

Use this function to remove the class driver and release the associated resources.

Format

```
t_usbd_tec_scanner_ret usbd_tec_scanner_delete ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBD_TEC_SCANNER_SUCCESS	Successful execution.
USBD_TEC_SCANNER_ERROR	Operation failed.

4.2 Device Management

The functions are the following:

Function	Description
usbd_tec_scanner_send_code()	Sends a bar code to the host.
usbd_tec_scanner_reg_ntf_fn()	Registers a notification callback function.

usbd_tec_scanner_send_code

Use this function to send a bar code to the host.

Format

```
t_usbd_tec_scanner_ret usbd_tec_scanner_send_code (
    uint8_t *   p_buffer,
    uint32_t    buf_len )
```

Arguments

Parameter	Description	Type
p_buffer	A pointer to the buffer to send. STX and ETX are not added.	uint8_t *
buf_len	The length of the buffer in bytes.	uint32_t

Return Values

Return value	Description
USBD_TEC_SCANNER_SUCCESS	Successful execution.
USBD_TEC_SCANNER_ERR_NOT_ENABLED	The host has not sent an 'Enable' command or the device has not enumerated yet.
USBD_TEC_SCANNER_ERR_BUSY	The last transfer has not finished yet.
USBD_TEC_SCANNER_ERR_TOO_LONG	*buf_len* is greater than USBD_TEC_SCANNER_MAX_BARCODE_LENGTH.
USBD_TEC_SCANNER_ERROR	Operation failed to set an event for the transfer task.

usbd_tec_scanner_reg_ntf_fn

Use this function to register a notification callback function.

Note: It is the user's responsibility to provide any callback functions the application requires. Providing such functions is optional.

Format

```
t_usbd_tec_scanner_ret usbd_tec_scanner_reg_ntf_fn ( t_usbd_tec_scanner_ntf_fn ntf_fn )
```

Arguments

Parameter	Description	Type
ntf_fn	The callback function.	t_usbd_tec_scanner_ntf_fn

Return Values

Return value	Description
USBD_TEC_SCANNER_SUCCESS	Successful execution.
USBD_TEC_SCANNER_ERROR	Operation failed.

4.3 Error Codes

If a function executes successfully, it returns with USBD_TEC_SCANNER_SUCCESS. The following table shows the meaning of the error codes.

Return Code	Description
USBD_TEC_SCANNER_SUCCESS	Successful execution.
USBD_TEC_SCANNER_ERR_RESOURCE	Resource (mutex, event, or task) allocation error.
USBD_TEC_SCANNER_ERR_NOT_ENABLED	The host has not sent an 'Enable' command or the device has not enumerated yet.
USBD_TEC_SCANNER_ERR_BUSY	The last transfer has not finished yet.
USBD_TEC_SCANNER_ERR_CODE_TOO_LONG	The buffer length is greater than USBD_TEC_SCANNER_MAX_BARCODE_LENGTH.
USBD_TEC_SCANNER_ERROR	Operation failed.

4.4 Types and Definitions

This section describes the main elements that are defined in the API Header file.

t_usbd_tec_scanner_ntf_fn

The **t_usbd_tec_scanner_ntf_fn** definition specifies the format of the notification function that can be called when a state change occurs on the control channel.

Format

```
typedef void ( * t_usbd_tec_scanner_ntf_fn )(
    uint8_t                     uid,
    t_usbd_tec_scanner_st_type  ntf )
```

Arguments

Parameter	Description	Type
uid	The unit ID. This is always 0.	uint8_t
ntf	The device state (the notification type).	t_usbd_tec_scanner_st_type

t_usbd_tec_scanner_st_type

The posssible device states are as follows:

Return Code	Description
USBD_TEC_SCANNER_ST_DISCONNECT	Device disconnected (this is the default state).
USBD_TEC_SCANNER_ST_CONNECT	A device is connected (USB enumeration complete).
USBD_TEC_SCANNER_ST_DISABLE	Command 'disable' received.
USBD_TEC_SCANNER_ST_ENABLE	Command 'enable' received; the bar code can be sent.
USBD_TEC_SCANNER_ST_RESET	Command 'reset' received.
USBD_TEC_SCANNER_ST_ERROR	A USB error occurred.

5 Integration

This section specifies the elements of this package that need porting, dependent on the target environment.

5.1 OS Abstraction Layer

All HCC modules use the OS Abstraction Layer (OAL) that allows the module to run seamlessly with a wide variety of RTOSes, or without an RTOS.

The TEC scanner module uses the following OAL components:

Resource	Requirement
Tasks	1
Mutexes	1
Events	1

5.2 PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer. For full details of its functions and macros, see the *HCC Base Platform Support Package User Guide*.

The module makes use of the following standard PSP functions:

Function	Package	Component	Description
psp_memcpy()	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.
psp_memset()	psp_base	psp_string	Sets the specified area of memory to the defined value.