UART Driver PSP User Guide

Version 1.10

For use with UART Driver PSP versions 3.01 and above

Date: 18-Jul-2017 13:46

Table of Contents

1 System Overview

1.1 Introduction

This guide is for those who want to do either of the following:

- use the HCC Embedded Platform Support Package (PSP) interface for Universal Asynchronous Receiver/Transmitter (UART) drivers from their application.
- implement a UART driver that conforms to the HCC Embedded UART PSP standard.

There are many different implementations of UART in microcontrollers. HCC has created a standard API for accessing UART drivers, so that all higher level software can be written independently of the specific UART hardware implementation. All HCC products that access a UART interface use the standard UART API interface defined in this document.

The UART PSP holds 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. The PSP provides the base functionality you need for your developments.

All PSP components are shipped as templates, code that you can use to build a real working PSP. In most cases real (default) code is included. The main components of the UART PSP template are PSP functions like **psp_uart_rx()** and **psp_uart_tx()**. This allows you to optimize these functions for your needs.

> **Note:** HCC provides a wide range of ported UART drivers, suitable for most standard microcontrollers, that conform to this standard.

1.2 Feature Check

The main features of the module are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Provides the standard UART Application Programming Interface (API) that is used by all HCC products that access a UART interface.

1.3 Packages and Documents

Packages

The table below 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.
psp_template_base	The base PSP package.
psp_template_uart	The UART PSP package described in this document.

Documents

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 Base Platform Support Package User Guide

This is the PSP base document.

HCC UART Driver PSP User Guide

This is this document.

1.4 Change History

This section describes past changes to this manual.

- To view or download earlier manuals, see Archive: UART Driver PSP User Guide.
- For the history of changes made to the package code itself, see History: psp_template_uart.

The current version of this manual is 1.10. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.10	2017-07-18	3.01	Added *Change History*. Added function group tables to API.
1.00	2015-04-02	3.01	First release.

2 Source File List

This section lists all the source code files included in the system. These files follow HCC Embedded's 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.

2.1 Platform Support Package (PSP) Files

These files are in the directory **src/psp**:

File	Description
include/psp_uart.h	Header file for UART functions.
target/uart/psp_uart_template.c	UART functions code file.

2.2 Version File

The file **src/version/ver_psp_uart.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 Application Programming Interface

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

The module management functions initialize, start, stop and delete the module. The remaining functions handle ISRs, reading and writing data. The template for callback functions is described.

3.1 Module Management

The functions are the following:

Function	Description
psp_uart_init()	Initializes a UART port and allocates the required resources.
psp_uart_start()	Starts a UART port.
psp_uart_stop()	Stops a UART port.
psp_uart_delete()	Deletes a UART port and releases the resources it used.

psp_uart_init

Use this function to initialize a UART port.

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

Format

```
t_psp_uart_ret psp_uart_init (
    const uint8_t                   uid,
    const t_psp_uart_dsc * const    p_dsc )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_dsc	A pointer to the UART descriptor.	t_psp_uart_dsc *

Return Values

Return value	Description
PSP_UART_SUCCESS	Successful execution.
PSP_UART_ERROR	Operation failed.

psp_uart_start

Use this function to start a UART port.

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

Format

```
t_psp_uart_ret psp_uart_start (
    const uint8_t         uid,
    const t_psp_uart_cb   cb,
    const uint32_t        cb_param )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
cb	The callback function.	t_psp_uart_cb
cb_param	The callback parameter.	uint32_t

Return Values

Return value	Description
PSP_UART_SUCCESS	Successful execution.
PSP_UART_ERROR	Operation failed.

psp_uart_stop

Use this function to stop a UART port.

Format

```
t_psp_uart_ret psp_uart_stop ( const uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_UART_SUCCESS	Successful execution.
PSP_UART_ERROR	Operation failed.

psp_uart_delete

Use this function to delete a UART port and release the associated resources.

Format

```
t_psp_uart_ret psp_uart_delete ( const uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_UART_SUCCESS	Successful execution.
PSP_UART_ERROR	Operation failed.

3.2 UART Management

The functions are the following:

Function	Description
psp_uart_isr_disable()	Disables RX and/or TX interrupts on a UART channel.
psp_uart_isr_enable()	Enables RX and/or TX interrupts on a UART channel.
psp_uart_rx()	Receives data over the UART.
psp_uart_tx()	Sends data over the UART.
psp_uart_tx_ch()	Writes one character over the UART.

psp_uart_isr_disable

Use this function to disable RX and/or TX interrupts on a UART channel.

Format

```
t_psp_uart_ret psp_uart_isr_disable (
    const uint8_t   uid,
    const uint8_t   flags )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
flags	The interrupt to disable (PSP_UART_FLAG_RX or PSP_UART_FLAG_TX).	uint8_t

Return Values

Return value	Description
PSP_UART_SUCCESS	Successful execution.
PSP_UART_ERROR	Operation failed.

psp_uart_isr_enable

Use this function to enable RX and/or TX interrupts on a UART channel.

Format

```
t_psp_uart_ret psp_uart_isr_enable (
    const uint8_t   uid,
    const uint8_t   flags )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
flags	The interrupt to enable (PSP_UART_FLAG_RX or PSP_UART_FLAG_TX).	uint8_t

Return Values

Return value	Description
PSP_UART_SUCCESS	Successful execution.
PSP_UART_ERROR	Operation failed.

psp_uart_rx

Use this function to receive data over the UART.

Format

```
t_psp_uart_ret psp_uart_rx (
    const uint8_t      uid,
    uint8_t * const    p_buf,
    const uint32_t     buf_len,
    uint32_t * const   p_read_len )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_buf	A pointer to the buffer to read into.	uint8_t *
buf_len	The buffer length.	uint32_t
p_read_len	Where to write the number of bytes read.	uint32_t *

Return Values

Return value	Description
PSP_UART_SUCCESS	Successful execution.
PSP_UART_ERROR	Operation failed.

psp_uart_tx

Use this function to send data over the UART.

Format

```
t_psp_uart_ret psp_uart_tx (
    const uint8_t      uid,
    uint8_t * const    p_buf,
    const uint32_t     buf_len,)
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_buf	A pointer to the buffer to send data from.	uint8_t *
buf_len	The buffer length.	uint32_t

Return Values

Return value	Description
PSP_UART_SUCCESS	Successful execution.
PSP_UART_ERROR	Operation failed.

psp_uart_tx_ch

Use this function to write one character over the UART.

Format

```
t_psp_uart_ret psp_uart_tx_ch (
    const uint8_t   uid,
    const uint8_t   ch )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
ch	The character to write.	uint8_t

Return Values

Return value	Description
PSP_UART_SUCCESS	Successful execution.
PSP_UART_ERROR	Operation failed.

3.3 Callback Functions

This callback function prototype is registered by **psp_uart_start()** and called by the UART driver when any PSP_UART_FLAG event occurs.

> **Note:** It is the user's responsibility to provide any callback functions required by the application. Providing such functions is optional.

Format

```
typedef void ( * t_psp_uart_cb )(
    const uint32_t   param,
    const uint8_t    uid,
    const uint8_t    flags,
    const uint8_t    b_isr )
```

Arguments

Parameter	Description	Type
param	The parameter provided by **psp_uart_start()**.	uint32_t
uid	The unit ID.	uint8_t
flags	The UART flag.	uint8_t
b_isr	1 if called from an ISR, 0 otherwise.	uint8_t

Return Values

Return value
None.

4 Types and Definitions

4.1 t_psp_uart_dsc

The **t_psp_uart_dsc** structure is defined as follows:

Element	Type	Description
baudrate	uint32_t	The required baud rate.
bits	uint8_t	The UART bits.
parity	uint8_t	The parity.
stop	uint8_t	The stop bits.

4.2 UART Flags

The UART flag definitions are as follows:

Definition	Value	Description
PSP_UART_FLAG_RX	1U	Data received.
PSP_UART_FLAG_TX	2U	Data written.
PSP_UART_FLAG_RXERR	4U	Data receive error.
PSP_UART_FLAG_TXERR	8U	Data write error.

4.3 Bit, Parity, and Stop Bit Definitions

UART Bits

These define the number of bits in each data word:

Option	Value	Meaning
PSP_UART_BITS_5	0U	5 data bits per word.
PSP_UART_BITS_6	1U	6 data bits per word.
PSP_UART_BITS_7	2U	7 data bits per word.
PSP_UART_BITS_8	3U	8 data bits per word.

Parity

The definitions are as follows:

Definition	Value	Description
LCT_PARITY_NONE	0U	No parity.
LCT_PARITY_ODD	1U	Odd parity.
LCT_PARITY_EVEN	2U	Even parity.
LCT_PARITY_MARK	3U	Mark parity.
LCT_PARITY_SPACE	4U	Space parity.

Stop Bits

The definitions are as follows:

Option	Value	Meaning
LCT_STOP_1	0U	One stop bit.
LCT_STOP_1_5	1U	One and a half stop bits.
LCT_STOP_2	2U	Two stop bits.