SPI Driver PSP User Guide

Version 1.20

For use with SPI Driver PSP versions 3.01 and above

Date: 18-Jul-2017 13:23

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 Serial Peripheral Interface (SPI) drivers from their application.
- implement an SPI driver that conforms to the HCC Embedded SPI PSP standard.

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

The SPI 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 SPI PSP template are PSP functions like **psp_spi_cs_hi()**, **psp_spi_rx()** and **psp_spi_tx()**. This allows you to optimize these functions for your needs.

> **Note:** HCC provides a wide range of ported SPI 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 a standardized abstraction of the SPI interface.
- Supports multiple SPI channels.
- Supports interrupt-driven SPI.
- Supports Direct Memory Access (DMA) SPI transfers.
- Compatible with all HCC products that use SPI.
- HCC provides a range of target-specific implementations for all standard micro-controllers.

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_spi	The SPI 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 SPI 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: SPI Driver PSP User Guide.
- There is currently no history of changes made to the package code itself.

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	2017-07-18	3.01	New *Change History* format.
1.10	2016-04-18	3.01	Added *Change History*. Added function group tables to API.
1.00	2014-12-04	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 PSP Files

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

File	Description
include/psp_spi.h	Header file for SPI functions.
target/spi/psp_spi.c	SPI functions code file.

2.2 Version File

The file **src/version/ver_psp_spi.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 SPI management functions handle chip select, baud rate, ISRs, (un)locking the SPI, reading and writing data.

3.1 Module Management

The functions are the following:

Function	Description
psp_spi_init()	Initializes an SPI port and allocates the required resources.
psp_spi_start()	Starts an SPI port.
psp_spi_stop()	Stops an SPI port.
psp_spi_delete()	Deletes an SPI port and releases the resources it used.

psp_spi_init

Use this function to initialize an SPI port.

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

Format

```
t_spi_ret psp_spi_init ( uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

psp_spi_start

Use this function to start an SPI port.

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

Format

```
t_spi_ret psp_spi_start ( uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

psp_spi_stop

Use this function to stop an SPI port.

Format

```
t_spi_ret psp_spi_stop ( uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

psp_spi_delete

Use this function to delete an SPI port and release the associated resources.

Format

```
t_spi_ret psp_spi_delete ( uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

3.2 SPI Management

The functions are the following:

Function	Description
psp_spi_cs_hi()	Sets chip select high.
psp_spi_cs_lo()	Sets chip select low.
psp_spi_get_baudrate()	Gets the baud rate.
psp_spi_set_baudrate()	Sets the baud rate.
psp_spi_lock()	Locks the SPI for the specific unit.
psp_spi_unlock()	Unlocks the SPI for the specific unit.
psp_spi_register_isr()	Registers the ISR ID for an SPI unit. This only applies if ISRs are supported.
psp_spi_rx()	Receives data over the SPI.
psp_spi_tx()	Transmits data over the SPI.
psp_spi_tx1()	Transmits one byte.
psp_spi_txrx()	Transmits and receives data over the SPI.

psp_spi_cs_hi

Use this function to set chip select high.

Format

```
void psp_spi_cs_hi ( uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value
None.

psp_spi_cs_lo

Use this function to set chip select low.

Format

```
void psp_spi_cs_lo ( uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value
None.

psp_spi_get_baudrate

Use this function to get the baud rate.

Format

```
t_spi_ret psp_spi_get_baudrate (
    uint8_t      uid,
    uint32_t *   p_br )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_br	The baud rate in Hz.	uint32_t *

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

psp_spi_set_baudrate

Use this function to set the baud rate.

Format

```
t_spi_ret psp_spi_set_baudrate (
    uint8_t    uid,
    uint32_t   p_br )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_br	The baud rate in Hz.	uint32_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

psp_spi_lock

Use this function to lock the SPI for the specific unit.

This can be useful if multiple units are attached to the same SPI bus.

Format

```
t_spi_ret psp_spi_lock (
    uint8_t   uid,
    uint8_t   from_isr )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
from_isr	The function is called from the interrupt flag.	uint8_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

psp_spi_unlock

Use this function to unlock the SPI for the specific unit.

This can be useful if multiple units are attached to the same SPI bus.

Format

```
t_spi_ret psp_spi_unlock (
    uint8_t   uid,
    uint8_t   from_isr )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
from_isr	The function is called from the interrupt flag.	uint8_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

psp_spi_register_isr

Use this function to register the ISR ID for an SPI unit.

> **Note:** This only applies if ISRs are supported.

Format

```
t_spi_ret psp_spi_register_isr (
    uint8_t        uid,
    oal_isr_id_t   isr_id )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
isr_id	The ISR ID.	oal_isr_id_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

psp_spi_rx

Use this function to receive data over the SPI.

Format

```
t_spi_ret psp_spi_rx (
    uint8_t     uid,
    uint8_t *   p_dst,
    uint32_t    len )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_dst	A pointer to the destination buffer.	uint8_t *
len	The number of bytes to receive.	uint32_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

psp_spi_tx

Use this function to transmit data over the SPI.

Format

```
t_spi_ret psp_spi_tx (
    uint8_t     uid,
    uint8_t *   p_src,
    uint32_t    len )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_src	A pointer to the source buffer.	uint8_t *
len	The number of bytes to transmit.	uint32_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

psp_spi_tx1

Use this function to transmit one byte.

Format

```
t_psp_spi_ret psp_spi_tx1 (
    uint8_t   uid,
    uint8_t   val)
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
val	The value to send.	uint8_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.

psp_spi_txrx

Use this function to transmit and receive data over the SPI.

> **Note:** SPI transfers are always bi-directional: the master side always sends data to clock in received data. When this function is called, transmitting of the requested data causes the reception of the identical number of bytes.

Format

```
t_spi_ret psp_spi_txrx (
    uint8_t     uid,
    uint8_t *   p_src,
    uint8_t *   p_dst,
    uint32_t    len )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_src	A pointer to the source buffer.	uint8_t *
p_dst	A pointer to the destination buffer.	uint8_t *
len	The number of bytes to transmit/receive.	uint32_t

Return Values

Return value	Description
PSP_SPI_SUCCESS	Successful execution.
PSP_SPI_ERROR	Operation failed.