Qualcomm Technologies International, Ltd.

Confidential and Proprietary – Qualcomm Technologies International, Ltd.

(formerly known as Cambridge Silicon Radio Ltd.)

NO PUBLIC DISCLOSURE PERMITTED: Please report postings of this document on public servers or websites to: DocCtrlAgent@qualcomm.com.

Restricted Distribution: Not to be distributed to anyone who is not an employee of either Qualcomm Technologies International, Ltd.or its affiliated companies without the express approval of Qualcomm Configuration Management.

Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies International, Ltd.

Any software provided with this notice is governed by the Qualcomm Technologies International, Ltd. Terms of Supply or the applicable license agreement at https://www.csrsupport.com/CSRTermsandConditions.

Qualcomm is a trademark of Qualcomm Incorporated, registered in the United States and other countries. All Qualcomm Incorporated trademarks are used with permission. Other product and brand names may be trademarks or registered trademarks of their respective owners.

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited.

© 2015 Qualcomm Technologies International, Ltd. All rights reserved.

Qualcomm Technologies International, Ltd.
Churchill House
Cambridge Business Park
Cambridge, CB4 0WZ
United Kingdom

Push every boundary.™

CSR µEnergy®

SPI Master & Slave Example Test Setup

Application Note

Issue 1

© Cambridge Silicon Radio Limited 2014
Confidential Information - This Material is Subject to CSR's Non-disclosure Agreement

Document History

Revision	Date	History
1	31 JAN 14	Original publication of this document

Contacts

General information	www.csr.com
Information on this product	sales@csr.com
Customer support for this product	www.csrsupport.com
More detail on compliance and standards	product.compliance@csr.com
Help with this document	comments@csr.com

© Cambridge Silicon Radio Limited 2014
Confidential Information - This Material is Subject to CSR's Non-disclosure agreement

Trademarks, Patents and Licences

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by CSR plc and/or its affiliates.

Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to CSR.

Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc or its affiliates.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

Use of this document is permissible only in accordance with the applicable CSR licence agreement.

Safety-critical Applications

CSR's products are not designed for use in safety critical devices or systems such as those relating to: (i) life support; (ii) nuclear power; and/or (iii) civil aviation applications, or other applications where injury or loss of life could be reasonably foreseeable as a result of the failure of a product. The customer agrees not to use CSR's products (or supply CSR's products for use) in such devices or systems.

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.

CSR

Contents

Tables, Figures and Equations

1. Introduction

This document describes the test setup required to test the SPI Master and the SPI Slave example applications that are included in the CSR µEnergy SDK. The SPI Master communicates with the SPI Slave to exchange data. The SPI transfers are full duplex and so data is exchanged in both the directions simultaneously.

The SPI Master and the SPI Slave applications use the PIO configurations listed in Table 1.1 for the SPI signals.

Signal	SPI Master Application PIO	SPI Slave Application PIO
SSEL	12	12
SCLK	9	9
MOSI	10	10
MISO	11	11

Table 1.1: PIO Configuration

See the *Development Board Connector Pin Assignment* in the SDK Support Documentation for PIO pin assignments for each development board.

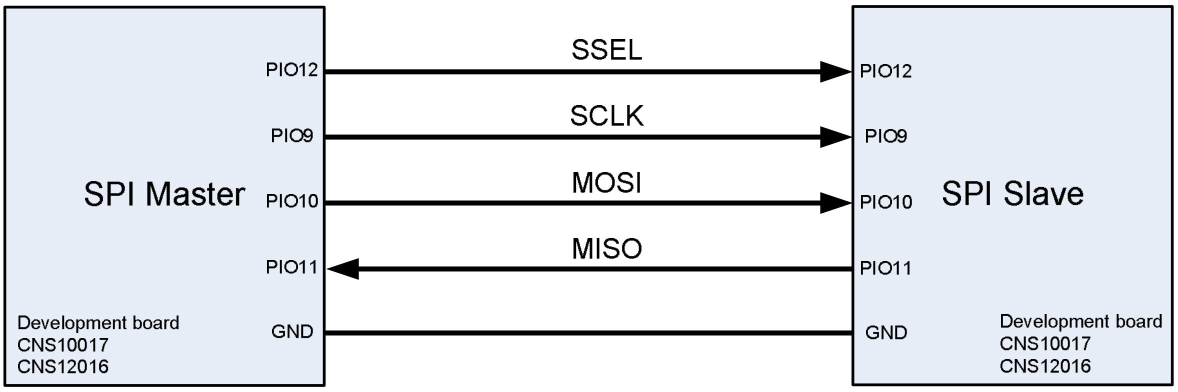

Figure 1.1: Hardware Setup

Two development boards are required and connected together as shown in Figure 1.1; one programmed with the SPI Master application and the other with the SPI Slave application.

Note:

The SPI Slave application must be started first and once running, the SPI Master can then be started, otherwise the SPI Master may encounter continuous SPI mismatches until the SPI Slave application starts running.

© Cambridge Silicon Radio Limited 2014
Confidential Information - This Material is Subject to CSR's Non-disclosure agreement

2. SPI Master Example Application

The SPI Master example application sends data of various lengths to the remote SPI Slave device. It expects the remote slave device to buffer the data and send the data back in the next transaction. After each transaction, the SPI Master compares the data received in the current transaction with the data sent in the previous transaction, and keeps a count of the total number of transactions and the number of matches. For each transaction, the SPI Master outputs a string summarising the transaction statistics onto the UART. For more information, see the *SPI Master Example Application Note*.

2.1. Prerequisites

This application requires:

- A CSR µEnergy CSR1001 development board (CNS10017), or a CSR µEnergy CSR1011 development board (CNS12016)
- A CSR µEnergy USB to SPI adapter board (CNS10020) plus mini-USB cable
- A CSR µEnergy 8-way adapter cable
- A host PC system with a USB connection running the CSR µEnergy SDK

To download the application to the development board, connect the USB cable to the PC, and the USB to SPI adapter to the development board using the 8-way adapter cable.

2.2. Software Requirements

In addition to the SDK installation, this application requires a terminal client e.g. HyperTerminal, PuTTY, etc.

2.3. Running the Application

1. Open the `spi_master` application workspace (`spi_master.xiw`) in xIDE. Select a build configuration (`Debug` or `Release`) from the drop-down field in the toolbar.
2. Specify which transport the device is connected to by opening the **Debug** menu and clicking on **Transport...**
3. Build (**F7**) the application.
4. Run (**F5**) the application.

2.4. Controlling the Application

Once executing, the application runs until stopped, reset or powered down. The application output can be viewed using the terminal client. This application does not accept any input from the user.

2.5. Configuration Details

Configure your terminal client to connect to the virtual UART exposed by the USB to SPI adapter board (CNS10020). The default terminal settings are 2400-8-N-1.

2.6. Output

In each transaction the SPI master:

- Asserts the SSEL
- Sends an octet indicating the data length (the length octet)
- Sends a fixed pattern of data of the specified length
- De-asserts the SSEL

Since the SPI transfers are full duplex, when the SPI Master sends data of a certain length, it also receives the data of the same length. The SPI Master application expects a remote SPI Slave device to buffer the data it received during one transaction, and to send the data back in a subsequent transaction. The SPI Master application then compares the data it received in a transaction with the data it has sent in the previous transaction, see Figure 2.1.

If the previous transaction was shorter than the current transaction, then only a portion of the received data would contain the data sent in the previous transaction.

© Cambridge Silicon Radio Limited 2014
Confidential Information - This Material is Subject to CSR's Non-disclosure agreement

If the previous transaction was longer than the current transaction, then data of only an amount equal to the length of the current transaction would be received. In this case the SPI Master application compares the received data with only a portion of the data sent in the previous transaction. Since the remote SPI Slave is unable to send all of the data it had buffered prior to the SSEL de-assertion, it discards the remaining data.

If the current transaction is of the same length as the previous transaction, then the SPI Master receives all of the data has sent in the previous transaction.

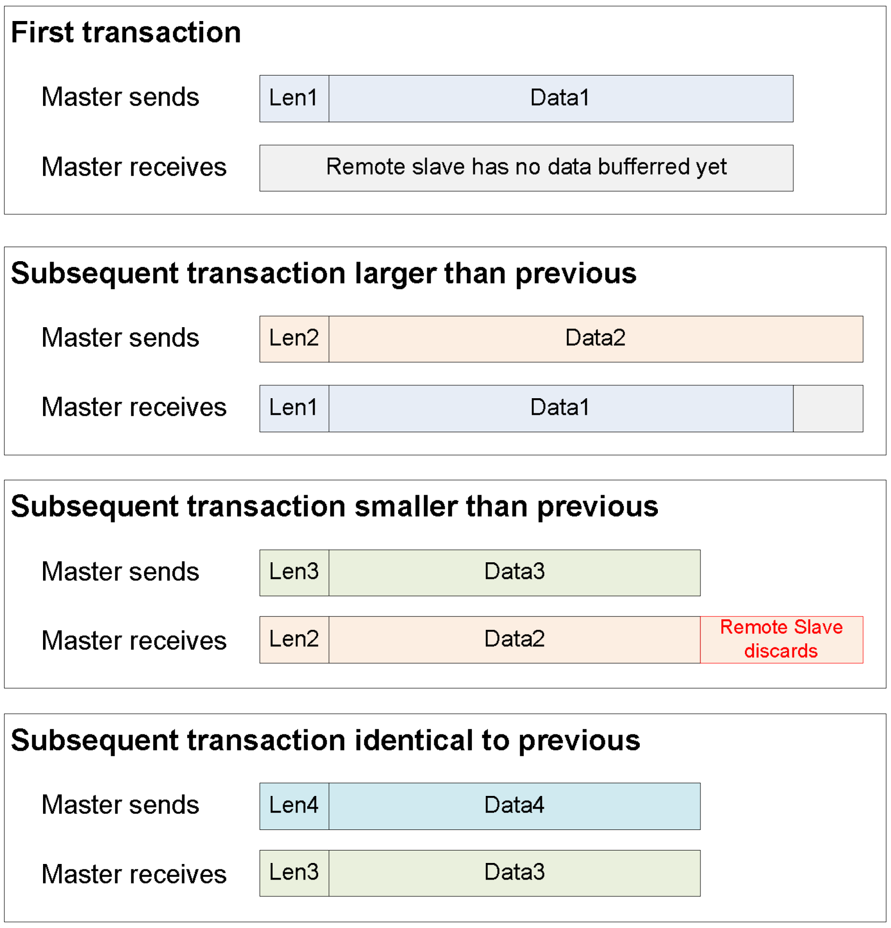

Figure 2.1: SPI Master Data Transactions

Whenever the SPI Master application completes a transaction, it outputs the accumulated statistics of the transactions so far.

If the length octet received does not match the length octet of the previous transaction, a mismatch in the length octet is reported. For example, the statistics would be output as:

```
Mismatch in length octet: Cumulative success rate 86% (118 matches out of
136 transactions)
```

If the length octet matches but the data does not match, a mismatch in the data is reported. For example, the statistics would be output as:

```
Mismatch in data received: Cumulative success rate 71% (191 matches out
of 269 transactions)
```

If both the length octet and the data match, then a successful transaction would be reported. For example, the statistics would be output as:

```
Successful transaction: Cumulative success rate 86% (119 matches out of
137 transactions)
```

© Cambridge Silicon Radio Limited 2014
Confidential Information - This Material is Subject to CSR's Non-disclosure agreement

3. SPI Slave Example Application

The SPI Slave example application simply buffers the data it received in a transaction, i.e. between the SSEL assertion and the de-assertion. It queues the received data up for transmission in a subsequent transaction. After a transaction, i.e. after the SSEL de-assertion, if any of the queued data is left over that data is simply discarded, prior to queuing up the newly received data for a subsequent transmission. For more information, see the *SPI Slave Example Application Note*.

3.1. Prerequisites

This application requires:

- A CSR µEnergy CSR1001 development board (CNS10017), or a CSR µEnergy CSR1011 development board (CNS12016)
- A CSR µEnergy USB to SPI adapter board (CNS10020) plus mini-USB cable
- A CSR µEnergy 8-way adapter cable
- A host PC system with a USB connection running the CSR µEnergy SDK

To download the application to the development board, connect the USB cable to the PC, and the USB to SPI adapter to the development board using the 8-way adapter cable.

3.2. Software requirements

In addition to the SDK installation, this application requires a terminal client e.g. HyperTerminal, PuTTY, etc.

3.3. Running the application

1. Open the `spi_slave` application workspace (`spi_slave.xiw`) in xIDE. Select a build configuration (`Debug` or `Release`) from the drop-down field in the toolbar.
2. Specify which transport the device is connected to by opening the **Debug** menu and clicking on **Transport...**
3. Build (**F7**) the application.
4. Run (**F5**) the application.

3.4. Controlling the Application

Once executing, the application runs until stopped, reset or powered down. The application output can be viewed using the terminal client. This application does not accept any input from the user.

3.5. Configuration Details

Configure your terminal client to connect to the virtual UART exposed by the USB to SPI adapter board (CNS10020). Default terminal settings are 2400-8-N-1.

© Cambridge Silicon Radio Limited 2014
Confidential Information - This Material is Subject to CSR's Non-disclosure agreement

3.6. Output

The SPI Slave application outputs the following characters:

Character Output	Description
`<`	SSEL is asserted
`L`	First octet containing the length has been received
`D`	Data is received
`>`	SSEL is de-asserted

Table 3.1: SPI Slave Application's Output Characters

Example output from the SPI slave application:

```
<LD><LD><L><LD><LD><LD><LD><LD><LD><LD><LD><L><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD>
<LD><LD><LD><LD><LD><LD><L><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD
><LD><LD><LD><LD><LD><L><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD><LD><L><LD
><LD><LD><LD><LD><LD>
```

© Cambridge Silicon Radio Limited 2014
Confidential Information - This Material is Subject to CSR's Non-disclosure agreement

Terms and Definitions

Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
Bluetooth Smart	Devices that support Bluetooth Low Energy
CSR	Cambridge Silicon Radio
e.g.	*exempli gratia* (for example)
i.e.	*id est.* (that is)
SDK	Software Development Kit
SPI	Serial Peripheral Interface
SSEL	Slave Select
SCLK	Serial clock
MOSI	Master Out Slave In (data output by master for slave to read in)
MISO	Master In Slave Out (data output by slave for master to read in)
PC	Personal Computer
PIO	Programmable Input/Output
SCLK	Serial clock
SDK	Software Development Kit
SPI	Serial Peripheral Interface
SSEL	Slave Select
UART	Universal Asynchronous Receiver Transmitter
xIDE	Integrated Development Environment for CSR µEnergy devices

© Cambridge Silicon Radio Limited 2014
Confidential Information - This Material is Subject to CSR's Non-disclosure agreement